UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

Detección de Fraude técnico en el sector de Telecomunicaciones

AROA RUIZ CASTELLO

Dirigido por

Dr. WOLFRAM ROZAS

CURSO 2024-2025

TÍTULO: Detección de Fraude técnico en el sector de Telecomunicaciones

AUTOR: AROA RUIZ CASTELLO

TITULACIÓN: MÁSTER BIG DATA

DIRECTOR/ES DEL PROYECTO: Dr. WOLFRAM ROZAS

FECHA: Septiembre de 2025

Índice General

Índice de Tablas

Índice de Figuras

Capítulo 1. RESUMEN DEL PROYECTO

En este capítulo sepresenta un resumen general del proyecto, donde se expone el contexto y la justificación que motivan su desarrollo, así como el planteamiento del problema y los objetivos que guían la investigación. Además, se incluyen de manera sintética los principales resultados alcanzados y se describe la estructura de la memoria, ofreciendo al lector una visión clara y ordenada de los contenidos que se desarrollan en los capítulos posteriores.

1.1 Contexto y justificación

El fraude en las telecomunicaciones es una amenaza persistente para el sector.

Las telecomunicaciones son una de las industrias más estratégicas y dinámicas a nivel global. Su rol en el desarrollo económico, social y tecnológico es fundamental, ya que conecta personas, empresas, gobiernos y dispositivos a través de voz, datos e infraestructura digital.

En las últimas décadas, las telecomunicaciones han experimentado una transformación profunda que ha revolucionado la forma en que nos comunicamos. Desde los servicios tradicionales de telefonía fija hasta las actuales redes móviles 5G, el acceso a internet de alta velocidad, las plataformas de mensajería instantánea y las tecnologías de voz sobre IP (VoIP), la industria telco ha sido clave para habilitar la conectividad global y digitalizar múltiples sectores económicos.

Este crecimiento exponencial ha traído consigo grandes beneficios, pero también nuevos riesgos. A medida que los servicios de telecomunicaciones se vuelven más accesibles y se diversifican, también lo hacen las oportunidades para los fraudes. El aumento en la complejidad de las redes, la integración de tecnologías como la inteligencia artificial, la nube y el IoT, y la presión por lanzar servicios rápidamente al mercado, han creado vulnerabilidades que pueden ser explotadas por agentes malintencionados.

Además, la digitalización de procesos internos, como la facturación electrónica, la portabilidad numérica, la atención al cliente automatizada y la contratación en línea, ha facilitado la comisión de fraudes sin contacto físico y a gran escala. Los delincuentes, muchas veces organizados en redes transnacionales, aprovechan vacíos legales, diferencias regulatorias entre países y la asimetría tecnológica entre operadores para desarrollar esquemas fraudulentos cada vez más sofisticados y difíciles de detectar.

En este escenario, las empresas de telecomunicaciones se enfrentan un doble desafío: por un lado, deben mantener su rentabilidad y asegurar la calidad del servicio en un entorno altamente competitivo; y por otro, deben protegerse frente a actividades fraudulentas que no solo generan pérdidas económicas significativas, sino que también pueden deteriorar la confianza de los usuarios y exponerlas a sanciones legales.

La lucha contra el fraude en telecomunicaciones se ha vuelto una prioridad estratégica para el sector. Por ello, comprender su origen, evolución, impacto y formas de prevención resulta esencial para las empresas, los reguladores y la sociedad en general.

El hecho de estudiar y analizar este tipo de fraude es crucial por varias razones:

1. Impacto financiero: Según la Asociación de Control de Fraude de las Comunicaciones (CFCA), el fraude en las telecomunicaciones ha incrementado un 12% solamente en 2023, provocando unas pérdidas estimadas en 38,95 billones de dólares. Estas cifras pueden comprometer la rentabilidad de muchas empresas del sector.
2. Vulnerabilidad del usuario final: Muchas formas de fraude afectan directamente al cliente, generando cargos indebidos, interrupciones en el servicio o incluso robo de identidad. Esto daña la experiencia del usuario y pone en riesgo su seguridad.
3. Operativas: Sobrecarga de los sistemas de atención al cliente y necesidad de implementar soluciones tecnológicas costosas.
4. Reputación de la industria: Los escándalos de fraude afectan la percepción pública de las telcos, minando la confianza y dificultando la fidelización de clientes.
5. Legales / Cumplimiento normativo: Las empresas están obligadas por leyes de protección de datos (GDPR) y derechos de los consumidores a prevenir y mitigar este tipo de riesgos. El incumplimiento puede acarrear sanciones y litigios.

Dada su magnitud e implicaciones, es necesario comprender a fondo este fenómeno para poder combatirlo de manera efectiva.

1.2 Planteamiento del problema

A pesar de los avances tecnológicos y de seguridad implementados por las empresas de telecomunicaciones, éstas continúan siendo blanco de múltiples formas de fraude**.** Los operadores, tanto grandes como pequeños, se enfrentan diariamente a intentos de estafa que no solo afectan sus ingresos, sino que también comprometen la calidad del servicio, la satisfacción del cliente y la integridad de sus sistemas. Esto evidencia deficiencias en los sistemas de control, la detección temprana y la cultura organizacional de prevención.

El problema se agrava debido a varios factores: por un lado, el ecosistema digital es cada vez más complejo, con múltiples dispositivos conectados, servicios en la nube, plataformas de autoservicio y una gran cantidad de transacciones automatizadas. Por otro lado, los defraudadores se adaptan rápidamente a los cambios tecnológicos, creando esquemas cada vez más elaborados que resultan difíciles de detectar con métodos tradicionales de monitoreo.

Muchas veces, las estrategias antifraude implementadas por las telcos son reactivas, costosas y poco eficaces frente a modalidades de fraude emergentes. A esto se suma que algunos operadores trabajan con sistemas antiguos (legados o "legacy") que no permiten una integración ágil de herramientas modernas de análisis y detección en tiempo real. En contextos donde el crecimiento del tráfico de datos y la expansión del mercado son prioridad, la prevención del fraude suele quedar en segundo plano, hasta que el impacto económico o reputacional se vuelve evidente.

Otro elemento que complica el escenario es la falta de estandarización y colaboración internacional. Como muchas operaciones fraudulentas cruzan fronteras —especialmente en

fraudes relacionados con el roaming, interconexión o tráfico VoIP—, los vacíos legales y la fragmentación regulatoria dificultan la persecución de los responsables y la implementación de soluciones coordinadas entre países.

En consecuencia, el problema que enfrentan las empresas de telecomunicaciones no es únicamente la existencia de fraude, sino la dificultad de anticiparlo, controlarlo y eliminarlo de manera eficiente y sostenible, sin afectar la operación normal ni incrementar los costos excesivamente.

En este contexto, surge la necesidad de abordar preguntas fundamentales como:

- ¿Cuáles son las formas más comunes y perjudiciales de fraude en las telecomunicaciones?
- ¿Por qué persiste el fraude, incluso en empresas con altos niveles de madurez tecnológica?
- ¿Qué impacto tiene el fraude en los ingresos, la reputación y la sostenibilidad de las telcos?
- ¿Qué estrategias de prevención y detección pueden implementarse para mitigar este riesgo de forma proactiva?

Responder a estas preguntas permitirá no solo dimensionar correctamente la magnitud del problema, sino también formular recomendaciones prácticas y estratégicas que contribuyan a fortalecer la ciberseguridad, la transparencia y la eficiencia en el sector de las telecomunicaciones.

En conclusión, el fraude en las telecomunicaciones representa un desafío complejo que requiere una combinación de tecnología avanzada, como sistemas de detección basados en inteligencia artificial, colaboración entre operadores y reguladores, y una mayor concienciación de los clientes. Las telcos deben adoptar un enfoque proactivo para mitigar estos riesgos, integrando estrategias de prevención, monitoreo en tiempo real y respuesta rápida para proteger sus operaciones y a sus usuarios.

1.3 Objetivos del proyecto

El objetivo general de este proyecto es diseñar e implementar un sistema de detección de fraude en telecomunicaciones que permita identificar de manera oportuna y precisa actividades fraudulentas dentro de la red, con el fin de minimizar pérdidas económicas, proteger la integridad del servicio y mejorar los mecanismos de control y prevención.

Los objetivos específicos se pueden ver en más detalle en el capítulo 3. Aquí una breve descripción:

1. Comprensión del Negocio (Business Understanding) - Estudio de las distintas modalidades de fraude y su impacto.
2. Comprensión de los Datos (Data Understanding) - Recopilación y análisis de datos

3. Preparación de los Datos (Data Preparation) - Limpieza y normalización de los datos.
4. Modelado (Modeling) - Evaluación de diferentes metodologías.
5. Evaluación (Evaluation) - Efectividad del modelo.
6. Despliegue (Deployment) - Implementación.

1.4 Resultados obtenidos

La investigación implementó un sistema híbrido de detección de fraude en telecomunicaciones utilizando tanto técnicas supervisadas como no supervisadas. En la primera etapa, se evaluaron cuatro modelos supervisados donde XGBoost demostró ser superior a Random Forest, Regresión Logística y SVC para la detección binaria de fraude. Posteriormente, se aplicó un enfoque no supervisado combinando K-means clustering, autoencoders e isolation forest para la clasificación de tipos de fraude.

El sistema logró una separación clara entre casos fraudulentos y normales, con scores promedio de -0.0292 para outliers y +0.1355 para casos normales, sin solapamiento entre distribuciones. Se identificaron dos tipos principales de fraude: el fraude sistemático organizado caracterizado por llamadas internacionales automatizadas de aproximadamente 20 minutos con costos uniformes de $60 ejecutadas principalmente a medianoche, y el fraude oportunista doméstico que involucra llamadas locales a destinos premium de 3-5 minutos con costos moderados de $8-16 concentradas en horarios de madrugada.

Los resultados revelaron patrones inesperados donde el 100% de los casos más críticos ocurrieron en horarios nocturnos, contrario a la hipótesis inicial de fraude internacional de alto costo, se encontró que el fraude es predominantemente doméstico con una reducción del 86% en llamadas internacionales entre los outliers. El sistema detectó uniformemente 5.01% de outliers por cluster con una distribución que permitió establecer umbrales operativos claros donde scores menores a -0.04 requieren revisión inmediata.

La metodología híbrida demostró efectividad tanto para la detección automática como para la clasificación precisa de diferentes tipos de fraude, proporcionando un sistema operacionalmente viable que puede automatizar el 75% del proceso de detección mientras mantiene alta precisión en la identificación de patrones tanto sistemáticos como oportunistas de actividad fraudulenta en redes de telecomunicaciones.

1.5 Estructura de la memoria

CAPÍTULO 2: ESTADO DEL ARTE

El fraude en telecomunicaciones representa pérdidas millonarias a nivel global y ha evolucionado junto con el desarrollo tecnológico, pasando de fraudes de facturación a ataques automatizados sobre vulnerabilidades en redes y sistemas. Históricamente, la detección ha transitado de reglas simples a enfoques modernos basados en machine learning, con avances hacia sistemas proactivos y en tiempo real. La literatura actual clasifica las técnicas en supervisadas, no supervisadas e híbridas, destacando métodos como Random Forest, SVM,

redes neuronales, clustering, autoencoders e isolation forest, cada uno con fortalezas y limitaciones. El problema abordado radica en superar retos como la alta tasa de falsos positivos, la dificultad de identificar nuevos fraudes, la incapacidad de clasificar automáticamente distintas modalidades y la necesidad de soluciones precisas pero interpretables.

CAPÍTULO 3: OBJETIVOS

El proyecto busca desarrollar un sistema híbrido de detección y clasificación de fraude en telecomunicaciones que combine algoritmos supervisados y no supervisados de machine learning para lograr alta precisión y caracterización automática de distintos fraudes. Para ello, se crea un dataset sintético representativo y se definen métricas específicas; se plantea comparar técnicas supervisadas en detección binaria yaplicar métodos no supervisados en clasificación. Los beneficios esperados incluyen la reducción de pérdidas económicas, la automatización del análisis de casos, la mejora en la experiencia del cliente al disminuir falsas alarmas, la posibilidad de replicar la metodología en otros operadores y el aporte científico en el área de detección de fraude con inteligencia artificial.

CAPÍTULO 4: DESARROLLO DEL PROYECTO

El proyecto se desarrolla bajo una metodología experimental que abarca revisión bibliográfica, creación de un dataset sintético, implementación y validación de algoritmos, con cronograma, criterios de éxito y gestión de riesgos definidos. La solución propuesta consiste en una arquitectura híbrida con una etapa supervisada basada en XGBoost —comparado con Random Forest, Regresión Logística y SVC— para detección binaria, y otra no supervisada que integra K-means, autoencoders e isolation forest para segmentación y detección de anomalías, complementada por un pipeline de datos y técnicas de feature engineering. Se requieren recursos tecnológicos (hardware de cómputo, librerías de machine learning y visualización) y humanos especializados, respaldados por un presupuesto que contempla desarrollo, implementación y mantenimiento. El análisis de viabilidad confirma la factibilidad técnica, económica y operativa de la propuesta, y los resultados experimentales muestran que XGBoost ofrece el mejor desempeño supervisado, mientras que las técnicas no supervisadas identifican eficazmente fraudes organizados y oportunistas, logrando métricas sólidas de separación, detección de outliers y umbrales operativos para priorizar casos.

Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE

En este capítulo 2 se aborda los antecedentes y el estado del arte relacionados con el fraude en telecomunicaciones. Se inicia con el contexto general, incluyendo aspectos clave sobre la facturación en el sector y los principales tipos de fraude. Posteriormente, se presentan los antecedentes con una comparativa geográfica que contrasta el panorama europeo con la situación particular de España. A continuación, se expone el estado del arte, donde se revisan las principales investigaciones y enfoques actuales en detección de fraude. Finalmente, se plantea el problema específico que da origen a este proyecto, destacando las limitaciones de las soluciones existentes y la necesidad de nuevas aproximaciones.

2.1 Contexto

Para poder entender el tema en cuestión deberemos analizar cómo es el proceso de facturación en las telco (proveedoras de servicios de telecomunicaciones) y a qué tipo de fraude se enfrenta.

2.1.1 Facturación en las telco

La facturación constituye un componente esencial en la industria de las telecomunicaciones, ya que asegura la correcta monetización de los servicios y contribuye a la satisfacción del cliente. Existen diversos modelos de facturación que se adaptan a distintos tipos de servicios y perfiles de usuarios, así como distintos ciclos de facturación que determinan la manera en que los clientes pagan por dichos servicios.

Modelos de facturación:

- Facturación Recurrente
  Es el modelo más usado en la industria de las telecomunicaciones.
  Se factura a los clientes una cantidad fija de modo regular, típicamente mensual o anual. Este modelo se adapta mucho mejor a servicios que tiene un uso consistente o es una oferta basada a suscripción.
  Este tipo de modelo tiene ciertas ventajas para ambos, proveedor y clientes. Para las operadoras se aseguran de unos ingresos periódicos, cosa que simplifican la planificación de las finanzas. Los clientes se benefician de la comodidad del pago automático.
- Facturación basada en uso
  Este modelo es otro de los preferidos en la industria de las telecomunicaciones. Se factura a los clientes basándose en la cantidad de servicios y recursos consumidos.

  La facturación basada en uso facilita flexibilidad y efectividad en el coste para ambos, el proveedor de servicios y sus clientes. Los clientes agradecen pagar solamente por lo servicios usados. Las operadoras se benefician de la posibilidad de escalar su beneficio dependiendo de las necesidades del cliente, resultando una facturación precisa y justa.
- Facturación de un solo uso

Este modelo factura al cliente por servicios o productos específicos, los cuales no son recurrentes.

Este modelo se usa para instalaciones o compras de dispositivos.

- Facturación escalonada
  La facturación escalonada ofrece diferentes niveles o paquetes de servicios con diversificación de funciones y precios. Los clientes pueden elegir que paquete se adapta mejor a sus necesidades y presupuesto, dándoles la flexibilidad de customizar los servicios de telecomunicaciones.
  Este modelo beneficia sobre todo a los proveedores de servicios con diversos tipos de clientes, ya que permite focalizar las ofertas en diferentes segmentos del mercado.

Ciclos de Facturación:

Además de los modelos de facturación, los proveedores de telecomunicaciones gestionan distintos ciclos de facturación según la modalidad de pago del cliente.

- Facturación Prepago: Pay-as-You-Go
  Esta solución es para clientes que quieran pagar por los servicios en antemano. La cantidad pagada se traduce en tiempo real en predeterminadas unidades, como minutos, datos o mensajes.
- Facturación Pospago:
  Esta solución es la más convencional, donde los clientes consumen ciertos servicios y son facturados al final de cada ciclo de facturación.
- Facturación Convergente
  Esta solución es más avanzada y compleja. Consolida múltiples servicios, servicios de voz, datos y contenidos, en una sola factura unificada.
  Debido a la irrupción de la tecnología digital y la creciente demanda de interacciones personalizadas con los clientes, el sector de las telecomunicaciones está experimentando una transformación que le lleva a incrementar el uso de la facturación convergente.
  Con la facturación convergente se reducen los errores de facturación: al facturar todos los servicios juntos es más fácil identificar y corregir errores.

Proceso de facturación:

El proceso de facturación en telecomunicaciones comprende las siguientes etapas:

Paso1 – Recolecta de datos y Agregación

El ciclo de facturación comienza recogiendo los archivos de uso de los servicios y la categorización de las llamadas. Los proveedores de servicio importan datos como transacción de mensajes, registros de los detalles de las llamadas (CDRs), y el ancho de banda usado, en sus sistemas de facturación. Se categorizan las llamadas con precisión, si son locales, intralata u on-network.

Paso 2 – Clasificación & Precios

Este es un paso fundamental, donde los cargos se calculan según tablas de tarifas predefinidas y los impuestos aplicables. Los sistemas de facturación relacionan cada CDR con el número de teléfono de origen para identificar al cliente y determinar el precio. Este proceso garantiza que cada llamada se calcule con precisión, reflejando el costo adecuado.

Los datos de llamadas recibidos son evaluados basándose en los planes de telecomunicaciones personalizados, permitiendo diferentes cargos basados en el tipo de llamada y su exceso.

Un sistema de clasificación flexible gestiona todo tipo de uso de telecomunicaciones, incluyendo minutos usados, excedentes, paquetes, llamadas gratuitas, larga distancia, internacional y SMS.

Paso 3 – Facturación y Generación de Facturas

En este paso el software de facturación, determinan tasas, redondea los decimales, y aplican minutos gratuitos si fuera aplicable. Durante la generación de las facturas, los sistemas calculan cargos recurrentes y no recurrentes, impuestos, descuentos y más. Las facturas detalladas incluyen información de contacto, descripciones de los cargos y resúmenes de servicios.

Paso 4 – Envío de la Factura

Paso 5 – Proceso de Pago

Paso 6 – Ajuste y Reembolsos

Paso 7 – Informes y Análisis

2.1.2 Tipos de Fraude

La complejidad de los modelos y ciclos de facturación en telecomunicaciones, si bien permite una gestión eficiente de los servicios y los ingresos, también abre la puerta a diversas formas de fraude. Estos fraudes pueden afectar tanto a los operadores como a los clientes, generando pérdidas económicas significativas y riesgos de seguridad. A continuación, se describen los principales tipos de fraude que se presentan en el sector, explicando sus mecanismos de operación y el impacto que pueden tener sobre los sistemas de facturación y la integridad del servicio.

Fraude internacional de reparto de ingresos (IRSF)

Este fraude toma ventaja de las tasas telefónicas Premium, que luego marcan los usuarios sin darse cuenta.

Es hasta ahora, el mayor desafío de fraude para los operadores de telecomunicaciones, costándole a la industria entre 4 y 6.1 mil millones de dólares al año.

Así es como funciona:

1. Los agentes maliciosos se registran para alquilar un número de teléfono premium.
2. Entran en los sistemas telefónicos de una empresa y hacen llamadas a ese número.
3. La empresa paga hasta $1 por minuto, el 25% del cual va a parar a los bolsillos del defraudador.

Las empresas pueden encontrarse repentinamente con facturas telefónicas astronómicas por llamadas que no reconocen. Las llamadas a menudo ocurren fuera del horario laboral y las empresas solo se dan cuenta de que se han realizado cuando llega el momento de pagar la factura.

Fraude de derivación de interconexión – Caja SIM (SIM BOX)

El fraude de derivación de interconexión, también conocido como fraude de caja SIM, toma ventaja de algo llamado tasa de terminación para hacer llamadas telefónicas más baratas. Se estima que cuesta a los operadores de telecomunicaciones 2700 millones de dólares en ingresos perdidos al año.

Para entenderlo, veamos un escenario con dos operadores en diferentes países.

- Un cliente del Operador A llama a un cliente del Operador B.
- El operador A cobra a su cliente una tasa por minuto.
- El Operador B le cobra al Operador A una tasa por proporcionar la llamada a su cliente.

Ese último cargo, donde la llamada termina, es la tasa de terminación. El problema es que estas tasas varían enormemente dependiendo de los contratos entre los dos operadores. Algunas de ellas son caras, otras están cerca de 0.

Aquí es donde entran en juego los expertos en fraudes de telecomunicaciones. Utilizan tarjetas SIM de un operador local y redirigen las llamadas internacionales usando una caja SIM o una puerta de enlace GSM. Ellos esencialmente están haciendo llamadas de larga distancia mucho más baratas para los que llaman y sacando dinero de los bolsillos de los operadores de telecomunicaciones.

Esto también afecta a la satisfacción del cliente. La mayoría de las veces, la calidad de estas llamadas es inferior a la de las llamadas internacionales estándar.

Fraude de arbitraje de telecomunicaciones

El arbitraje es la práctica general de capitalizar las diferencias de precios. En el mundo de las telecomunicaciones, estas diferencias aparecen en las tasas de larga distancia entre países.

Al igual que con el fraude de derivación internacional, puede reducir el costo internacional para los clientes, pero también abrir la puerta a empresas fraudulentas que se interponen entre los

operadores. Afirman conectarse directamente desde el país A al B, cuando de hecho, pasan por un país con una tasa más barata para conectar la llamada.

Hackeo PBX

El hackeo PBX permite a los defraudadores tomar el control de las líneas telefónicas mediante la explotación de redes telefónicas no seguras.

Un PBX o intercambio de sucursal privada (por sus siglas en inglés) es una red telefónica privada que se conecta a redes externas. Es lo que permite a las empresas compartir líneas y reducir la cantidad de números necesarios en una oficina, por ejemplo.

Debido a que muchos de estos PBX están basados en IP, pueden ser un objetivo fácil para los hackers. Iniciarán sesión en el sistema y lo utilizarán para su beneficio, por ejemplo, para el fraude IRSF mencionado anteriormente. Este es un problema de ciberseguridad e informática que se puede evitar con mejores controles internos y seguridad de las contraseñas.

Bombeo de tráfico

El bombeo de tráfico, también conocido como estimulación de acceso, es una práctica en la que las centrales locales sin escrúpulos manipulan el número de llamadas a sus redes para beneficiarse con las tasas de compensación establecidas por la Comisión Federal de Comunicaciones del país en cuestión.

Por ejemplo, en EEUU, según la Ley de Telecomunicaciones de 1996, las grandes empresas de telecomunicaciones como Sprint, Verizon y AT&T tienen que pagar tasas a los operadores rurales. Estos operadores hacen todo lo que está a su alcance para aumentar la cantidad de llamadas y obtener mayores pagos.

Fraude de depósito

El fraude de depósito apunta a las tiendas en línea de los operadores de telecomunicaciones utilizando números de tarjetas de crédito robadas. Los defraudadores suelen comprar tarjetas SIM de prepago, pero la misma técnica se aplica a los dispositivos (smartphones, routers, etc.).

El problema, por supuesto, es que las tiendas en línea de los operadores de telecomunicaciones son responsables de reembolsar las tasas en forma de devoluciones de cargos. Por supuesto, podrías confiar en tu procesador de pagos para reducir estas tasas o implementar herramientas de prevención de fraude por devolución de cargos. Esto hace que tu empresa sea vulnerable a altas tasas de falsos positivos cuando se bloquean clientes legítimos.

También existe una amenaza creciente en forma de redes proxy 4G, una práctica que ha estado en auge en los últimos años, tanto para uso comercial legítimo como residencial.

Fraude de subscripción

El fraude de suscripción en el mundo de la telefonía hace que los delincuentes firmen contratos utilizando identificaciones y números de tarjetas de crédito robados.

Los contratos telefónicos son más difíciles de comprar para los defraudadores que los artículos porque implican una forma de verificación KYC (Conozca a su cliente). Es decir, debes verificar la identidad del usuario antes de que puedan suscribirse. A los defraudadores les encantan los smartphones de alta gama que pueden adquirir a través de contratos. Es un caso simple de presentar identificaciones falsas, liberar el dispositivo y revenderlo en mercados de segunda mano. Cuando llega la compañía de reposiciones, se da cuenta de que la persona no existe.

Los defraudadores pasan las verificaciones KYC ya que tienen una gran cantidad de identidades robadas de dónde elegir, ya sea adquiridas a través de técnicas de phishing, compradas en la dark web o alquiladas a mulas de identificación.

Robo de identidad

Las empresas de telecomunicaciones que ofrecen cuentas de usuario en línea pueden ser víctimas de ataques de robo de identidad (ATO), donde los defraudadores encuentran los datos del nombre de usuario y contraseña de otros usuarios, e inician sesión en su lugar.

Smishing/Phishing de SMS

El Smishing, también conocido como Phishing de SMS, es la práctica de enviar SMS masivamente para obtener información personal de la persona que recibe los mensajes.

Las campañas de spam masivo son la perdición de los clientes y de las empresas de telecomunicaciones. Por ello, las redes de phishing de SMS se han vuelto expertas en evitar su detección. Se sabe que utilizan software para confirmar que los números a los que ellos apuntan son móviles y no de teléfonos fijos (así las empresas de telecomunicaciones no notan los indicadores de alerta), crean tiendas automáticas para revender los datos robados, e incluso proveer su propio servicio de hosting para alojar sitios y mercados de phishing.

El punto es que el volumen de phishing de SMS superó excesivamente más del 328% en el 2020, y mientras los operadores de telecomunicaciones no siempre se hacen cargo de ello, deberían sentirse incómodos al saber que sus compañías son accidentalmente cómplices en la práctica. Un simple sistema para monitorear los inicios de sesión y las transacciones viniendo de algún servicio B2B debería ser suficiente para asegurar que tu compañía de telecomunicaciones no está ayudando a los negocios de smishing.

Fraude Wangiri

Del japonés que significa “uno y corta”, el fraude de telecomunicaciones Wangiri consiste en despertar curiosidad en los clientes al llamarlos, dejando que el teléfono suene una vez, y

colgando. El cliente devolverá la llamada, llamando sin querer a un número costoso premium que el defraudador controla.

Una variante SMS también existe, donde los defraudadores envían un mensaje solicitando a los clientes que devuelvan la llamada a cierto número. Los típicos indicadores de alerta para este tipo de fraude de telecomunicaciones son picos en el tráfico a destinos de alto costo, que las empresas de telecomunicaciones deberían ser capaces de monitorear con sus sistemas internos.

La clave aquí está en saber que, como un negocio, deberías vigilar cuales son los números que son marcados automáticamente. Esto no es solo para las empresas de telecomunicaciones, de hecho, cualquier compañía donde las llamadas son una parte importante de la búsqueda de clientes potenciales o del servicio de atención al cliente podría ser una buena idea implementar una sencilla herramienta de búsqueda inversa de teléfonos para protegerse.

Intercambio de SIM / SIM swapping

El intercambio de SIM, también conocido como SIM swapping, consiste en que los defraudadores toman control de los SMS y llamadas de una persona al intercambiar un número telefónico por otro que ellos controlan.

Dado que cada vez más servicios en línea utilizan OTP (contraseña de una sola ocasión) y 2FA (doble factor de verificación) a través de llamadas y SMS, los defraudadores intentan tomar el control de los números de teléfono de las personas.

La manera en que lo hacen es a través de una forma de robo de identidad llamado Robo de SIM o Intercambio de SIM. Se ponen en contacto con el servicio de atención al cliente de la empresa de telecomunicaciones y solicitan transferir su número a una nueva SIM, que ellos controlan. Cuando el procedimiento está completo, ellos pueden recibir todas las OTPs y verificaciones SMS que necesiten para hackear las cuentas del cliente, desde las redes sociales hasta los bancos.

Los operadores de telecomunicaciones se han vuelto mejores mitigando este tipo de riesgo en los últimos años, simplemente implementando su propio 2FA o MFA (sistema de autenticación multifactor) para confirmar si el usuario solicitó legítimamente un cambio de número.

CLI Spoofing – Suplantación del número llamante

El CLI Spoofing consiste en alterar el Caller Line Identification (CLI) para hacer que una llamada parezca provenir de otro número.

El mecanismo utilizado es la manipulación del campo de origen en el protocolo SIP o SS7.

Este tipo de fraude es muy frecuente en campañas de vishing o estafas telefónicas (phishing de voz).

Fraude Roaming

En este tipo de fraude se hace un uso indebido de una tarjeta SIM en itinerancia (roaming) internacional para realizar o recibir tráfico, sin intención de pagar.

El atacante aprovecha el retraso en la facturación entre operadores para acumular minutos de tráfico antes de que se detecte.

FAS (False Answer Supervision)

Esta técnica marca una llamada como "respondida" antes de serlo realmente, generando cargos fraudulentos.

El defraudador manipula la señalización SS7 o SIP para iniciar la tarificación anticipadamente.

2.2 Antecedentes

Existen múltiples definiciones de fraude, aunque de manera general podemos decir que el fraude es simplemente cualquier actividad que conlleva obtener una ventaja financiera o causar perdidas mediante engaño, ya sea implícito o explicito. En el sector de las telecomunicaciones, el fraude se ha consolidado como una amenaza significativa, generando pérdidas millonarias para los operadores cada año.

Gran parte de este costo es asumido por las compañías, que a menudo optan por no integrar sistemas complejos de gestión de riesgos en sus arquitecturas, debido a los elevados costes y a la complejidad operativa que implica. Además, la estructura del sector, en la que los servicios se fragmentan y revenden a otras redes locales u operadores, facilita la aparición de prácticas fraudulentas que pueden ser difíciles de detectar y controlar.

En los apartados siguientes se abordará un análisis comparativo del fraude en telecomunicaciones a nivel geográfico, destacando las tendencias y patrones en diferentes regiones, con especial atención al panorama europeo y a la situación particular en España. Este enfoque permitirá comprender mejor la magnitud del problema e identificar los tipos de fraude más frecuentes en distintos mercados.

2.2.1 Comparativa geográfica

El análisis geográfico del fraude en telecomunicaciones permite identificar no solo la magnitud del problema en diferentes regiones, sino también las tipologías más frecuentes y los mercados más vulnerables.

Si se hace una comparativa geográfica del fraude vemos que:

- En 2023, se estimaron pérdidas globales de 38,95 mil millones USD por fraude en telecom —con tipologías como fraude de suscripción, PBX, entre otros.

- Operadores en África y Oriente Medio reportan pérdidas superiores al 20 % de sus ingresos, mientras que Europa occidental ronda el 7 % y Norteamérica cerca del 13 %.
- Según CFCA entre 2013 y 2017, los principales países donde tiene origen el fraude internacional fueron EE.UU. y España (6 %), Reino Unido y Rusia (5 %).
- En el mismo periodo, los países más afectados (terminación de llamadas fraudulentas) incluyeron Cuba, Letonia, Lituania, Reino Unido y Túnez.

Si analizamos las tendencias regionales, según Subex (2024):

Región	Tipos de fraude predominantes
APAC	SIM-box, SMS-phishing, PBX hacking, Wangiri
África	SIM-box, SIM-swap
MENA	CLI spoofing, fraudes por dispositivo
Europa	Manipulación de número (A-Number), fraude mayorista, vishing
Norteamérica	Robocalls, takeovers de cuenta

Tabla 1 - Tendencias Regionales (Subex 2024)

Si hacemos una tabla comparativa con las estimaciones de perdida por región según los datos de CFCA obtenemos:

Región	Pérdidas (%) sobre ingresos
África/MENA	>20%
Asia	≈ 19 %
Latam	>15%
Norteamérica	≈ 13 %
Europa Occidental	≈ 7 %
Europa Central/Este	≈ 8 %

Tabla 2 - Comparación por región (CFCA)

Analizando estos datos podemos concluir:

1. Impacto desigual: África, MENA y Asia muestran los ratios más altos de pérdidas (15-20 + %), mientras que Europa occidental tiene impactos menores (~7 %).

2. Diversidad tipológica: Cada región enfrenta amenazas especialmente adaptadas a su infraestructura (SIM-box en APAC/África, robocalls en Norteamérica, manipulación de número en Europa).
3. Origen y terminación: Países desarrollados como EE.UU. y Reino Unido también orquestan o terminan un volumen significativo de fraude internacional.
4. Necesidad de enfoque geolocalizado: Las soluciones eficientes deben abordar las modalidades y el ecosistema de fraude específico de cada región.

2.2.2 Panorama Europeo

En Europa, el fraude en telecomunicaciones se manifiesta a través de diversas modalidades que afectan tanto a operadores como a usuarios finales, generando pérdidas económicas significativas y desafíos operativos complejos.

Los fraudes más comunes son los siguientes:

1. SIM-Swapping:

- ENISA reporta que casi la mitad de los operadores europeos sufrieron incidentes de SIM-swap en los últimos 12 meses.
- Recomendaciones: fortalecer autenticación del usuario, revisar procesos del personal de atención y coordinar con bancos.

2. Fraude SIM-Box y IRSF

- El fraude SIM-box provoca un incremento del 700 % en estafas por roaming (2023-2028).
- Empresas como Subex muestran detección ML con tasas de éxito del 98 % y reducción de pérdidas en 60 %.
- En Europa central, entre 2 % y 6 % de los ingresos mayoristas se pierden por IRSF.

3. CLI Spoofing y PBX Hacking

- Europol estima que el fraude por vishing y desvío de llamadas supera los €29 mil millones anuales.

4. Fraude Roaming

- Europol resalta que PBX hacking, IRSF y fraudes por roaming siguen siendo amenazas técnicas clave, absorbiendo €10.6 mil millones en 2019.

2.2.3 Situación en España

En España, los operadores de telecomunicaciones enfrentan pérdidas significativas relacionadas con modalidades de fraude como SIM swap, SIM box, CLI spoofing y hackeo de PBX. Para abordar estas amenazas, el Ministerio de Transformación Digital ha promovido diversas medidas, incluyendo el bloqueo de llamadas con CLI falso, el registro de SMS y el apoyo a la implementación de un sistema nacional STIR/SHAKEN. Guiados por las recomendaciones de

ENISA, los operadores han reforzado los controles en los procesos de SIM swap y se fomenta la colaboración estrecha entre bancos, operadores y autoridades. Además, proyectos financiados con fondos europeos, como SCAMSTOP «Scams and fraud detection in Voice over IP networks», se han diseñado para contrarrestar las vulnerabilidades de las tecnologías de Voz sobre IP, aun relativamente recientes y susceptibles a ataques sofisticados.

En cuanto a las soluciones técnicas aplicadas en España y Europa, destacan:

- SIM-swap: autenticación reforzada, monitorización de portabilidad de número (APIs con bancos).
- SIM-box: uso de ML con Geo-filtering y detección en tiempo real, alcanzando 98 % de éxito.
- CLI spoofing / vishing: adopción de STIR/SHAKEN, control de identidades.
- PBX hacking: filtrado de rutas, firewalls de señalización SS7/SIP, autenticación de troncos internacionales.

En conclusión, España comparte los mismos retos técnicos que el resto de Europa, con especial incidencia en SIM swap, SIM box y CLI spoofing. Aunque persisten brechas y cierta demora tecnológica, el país está avanzando en la implementación de medidas robustas impulsadas por la UE y ENISA. La combinación de detección activa, machine learning y colaboración intersectorial constituye la estrategia principal para reducir las pérdidas por fraude, mientras que el desafío actual es acelerar la adopción completa de STIR/SHAKEN y fortalecer la infraestructura frente a CLI spoofing y PBX hacking.

2.3 Estado del Arte

El fraude en telecomunicaciones y las estrategias de mitigación asociadas han sido objeto de estudio por numerosos investigadores y organismos internacionales. La literatura existente aborda tanto la identificación y clasificación de los distintos tipos de fraude, como el desarrollo de metodologías y tecnologías para su detección y prevención, incluyendo técnicas de machine learning, análisis de tráfico en tiempo real y protocolos de autenticación avanzados.

A continuación, se presenta una tabla resumen con los estudios y publicaciones más relevantes para este trabajo, destacando los enfoques metodológicos, los hallazgos principales y las contribuciones al conocimiento sobre la prevención y gestión del fraude en telecomunicaciones. Esta revisión permite situar el presente estudio dentro del marco académico y tecnológico existente, identificando tendencias, vacíos y oportunidades de mejora en la investigación actual.

Artículo	Año	Metodología Principal	Tipo de Fraude	Resultados Clave	Beneficios	Evaluación
Fraud Detection in Telecommunications: History and Lessons Learned	2010	Métodos estadísticos y ML	Fraude en llamadas y facturación	Evolución de técnicas y lecciones importantes	Proporciona un marco histórico y recomendaciones para mejorar detección	Herramientas visuales
Telecommunications Fraud Machine Learning-based Detection	2023	Modelos de ML para detección basada en comportamiento	Fraude de llamadas internacionales y de suscripción	Alta precisión con ML supervisado	Mejora la detección temprana y reduce pérdidas financieras	Matriz de confusión, métricas como precisión, recall y F1-score
A Study of Fraud Types, Challenges and Detection Approaches in Telecommunication	2020	Revisión sistemática y propuesta de clasificación de fraudes	Fraude de suscripción, falso cobro, phishing	Identificación de desafíos y métodos existentes	Clarifica retos técnicos y orienta futuras investigaciones	Métricas como precisión, Recall, F1-score, AUC
An improve fraud detection framework via dynamic representations and adaptive frequency response filter (DPGFD)	2025	Multi-LSTM, atención, filtro adaptativo en grafos	Fraude camuflado y heterofilia en grafos sociales	Mejora +5% en AUC y F1 vs. estado del arte	Mejora detección de fraudes colaborativos y ocultos	métricas AUC, Recall y F1-score
FAME: Fraud Analytics using ML & Big Data for Telecom	2023	Minería de datos auto adaptativa + Big Data	Fraude de participación en ingresos internacionales (IRS)	Falsos positivos <5%, alta precisión y escalabilidad	Reduce pérdidas millonarias y mejora eficiencia operativa	Matriz de confusión

Tabla 3 - Resumen Estudios Relevantes

Fraud detection in telecommunications: History and lessons learned

Richard A. BECKER, Chris VOLINSKY, and Allan R. WILKS - 2010

Este artículo revisa más de una década de experiencia en detección de fraude en telecomunicaciones por parte de AT&T, analizando las estrategias, herramientas estadísticas y enfoques organizacionales más efectivos. Se centra en los fraudes internacionales (como la interconexión de llamadas ilegales), tarjetas telefónicas robadas, y manipulación de tarifas.

Metodología

- Data mining & estadística aplicada: El equipo utilizó modelos estadísticos supervisados, análisis exploratorio y data mining para detectar patrones inusuales en llamadas, tiempos, rutas y volúmenes.
- Análisis de outliers: Se identificaban desviaciones en llamadas por duración, destino o volumen a través de métodos de control estadístico de calidad (SPC) y árboles de decisión.
- Modelos de clasificación supervisada: Aplicaron modelos como regresión logística, árboles CART y redes bayesianas sobre conjuntos de datos etiquetados como fraude/no fraude.
- Visualización de datos: Herramientas visuales eran clave para facilitar a los analistas humanos detectar rápidamente anomalías en patrones de tráfico.
- Retroalimentación humana: La experiencia humana jugó un papel esencial para revisar casos detectados y alimentar los sistemas automáticos, creando un bucle de mejora continua.
- Pipeline en tiempo real: Se desarrollaron sistemas que analizaban streaming de datos con latencias mínimas (menores a minutos), alertando sobre comportamientos sospechosos.

Resultados

- La implementación de estos sistemas redujo el tiempo medio de detección de fraudes de semanas a horas.
- Se recuperaron millones de dólares anualmente gracias a la detección temprana.
- El enfoque híbrido hombre–máquina resultó ser más efectivo que soluciones 100 % automáticas.
- Mostraron que la interpretabilidad de los modelos es crítica en entornos de alto riesgo operativo.

Telecommunications Fraud Machine Learning-based Detection

Nazih Salhab & Paris-Est Sup - 2023

El artículo presenta una visión general de los tipos de fraude en telecomunicaciones (como IRSF, SIM Box y CLI spoofing) y evalúa distintos modelos de Machine Learning para detección automática de esos fraudes.

Metodología

- Revisión de técnicas de detección: Encuesta del estado del arte, cubriendo métodos basados en ML incluidos Random Forest, SVM y Gradient Boosting.
- Dataset de casos reales: Utilización de logs de llamadas (CDRs) con distribución altamente desequilibrada entre casos legítimos y fraudulentos.
- Ingeniería de características: Extracción de variables como duración de llamada, frecuencia de destinos, cambio rápido de ruta, etc.
- Entrenamiento de clasificadores: Evaluación de múltiples modelos ML: Random Forest \ Gradient Boosting \SVM
- Evaluación comparativa: Uso de matriz de confusión y métricas como precisión, recall y F1-score. También aplicaron técnicas de balanceo (ej. SMOTE) para manejar el desbalance.
- Análisis offline y online: Evaluación en batch y en flujo continuo de datos (simulando entorno real).

Resultados

- Rendimiento de los modelos: Random Forest y Gradient Boosting superaron a SVM, logrando alta precisión y recall.
- Manejo del desbalance: El uso de SMOTE mejoró significativamente la detección de fraudes minoritarios.
- ML en tiempo real: Mostraron que los modelos son aptos tanto para análisis offline como para despliegue en sistemas en línea, manteniendo robustez y eficiencia.

A Study of Fraud Types, Challenges and Detection Approaches in Telecommunication

Zhiyuan Chen - 2020

Este artículo realiza una revisión profunda de tres aspectos clave en el ámbito del fraude en telecomunicaciones:

1. Categorías de fraude
2. Desafíos técnicos en su detección
3. Enfoques y métodos técnicos para combatirlos

Su propósito es proporcionar una visión global tanto de la problemática como de las soluciones propuestas hasta la fecha.

Metodología

- Revisión sistemática de literatura
- Clasificación de tipos de fraude

- Identificación de desafíos
    - Gran volumen y alta dimensionalidad de datos (CDRs),
    - Desequilibrio de clases entre eventos legítimos y fraudulentos,
    - Necesidad de detección en tiempo real (~streaming),
    - Privacidad y legalidad en tratamiento de datos,
    - Adaptabilidad de modelos frente a fraudes mutantes.
- Revisión de técnicas y métricas
    - Herramientas:
        - Estadística tradicional (modelos de control, pruebas de hipótesis),
        - Machine Learning (SVM, redes neuronales, clustering, LDA + KL Divergencia para detección),
        - Minería de reglas (extraer patrones predefinidos),
        - Enfoques híbridos combinando aprendizaje supervisado y no supervisado.
    - Métricas sugeridas: Precision, Recall, F1-score, AUC, junto con criterios de costos asociativos por falsos positivos/negativos.
- Propuestas de guía técnica
    - Selección de variables discriminantes,
    - Aplicación de reducción de dimensionalidad,
    - Técnicas de balanceo (SMOTE, submuestreo),
    - Implementación de pipelines para análisis en tiempo real.

<u>Resultados</u>

- Clasificación robusta de tipos de fraude, facilitando elección de técnicas adecuadas.
- Identificación de los principales retos en el tratamiento de datos y modelado.
- Comparativa técnica de métodos eficaces con recomendaciones metodológicas y métricas para evaluaciones futuras.

An improve fraud detection framework via dynamic representations and adaptive frequency response filter

Juncheng Yang, Shuxia Li, Zijun Huang & Junhang Wu - 2025

Este technical paper propone un framework novedoso para la detección de fraude técnico en telecomunicaciones, que aborda la naturaleza dinámica del comportamiento de usuarios y resuelve los desafíos del camuflaje y la heterofilia presentes en los grafos de interacción.

Metodología

- Multi-LSTM Behavior Encoder: Codifica la evolución temporal del comportamiento de usuarios en múltiples escalas. Esto permite capturar patrones dinámicos en secuencias de uso (llamadas, mensajes, etc.)
- Attention-based Fusion Module: Funde las representaciones multiescalares generadas por los LSTM, asignando pesos adaptativos según su relevancia en la identificación de comportamiento anómalo
- Latent Synergy Network (LSN) Extractor: Construye un grafo de segundo orden que incorpora interacciones entre usuarios —tanto fraudulentos como normales— para capturar estructuras sociales y patrones colaborativos
- Adaptive Frequency-Filter Graph Learning: Introduce un filtro configurable por canal de frecuencia para manejar la heterofilia (usuarios conectados con nodos desiguales) y el camuflaje mediante sobre-suavización en grafos. Ajusta selectivamente componentes de alta y baja frecuencia en el proceso de agregación.

Resultados

- Mejora de rendimiento promedio: +5 % en métricas AUC, Recall y F1-score frente a los modelos comparados
- Los fraudes detectados presentan heterofilia en redes sociales escondiéndose entre usuarios normales.
- Los filtros adaptativos correlacionan la pérdida de información inducida por técnicas de suavizado en grafos tradicionales

FAME: Fraud Analytics using ML & Big Data for Telecom

Sudarson Roy Pratihar, Subhadip Paul, Pranab Kumar Dash & Amartya Kumar Das - 2023

El artículo aborda el problema del fraude en la industria de telecomunicaciones, que representa una pérdida global de 46.3 mil millones de USD anuales. Los enfoques tradicionales basados en reglas han mostrado una eficiencia limitada debido a la rápida evolución de los patrones de fraude. Por ello, se propone una solución industrializada que combina técnicas de minería de datos auto adaptativas con tecnologías de Big Data para detectar fraudes y descubrir nuevos patrones de manera precisa, eficiente y rentable.

Metodología

- Técnica de minería de datos auto adaptativa: Se implementa un enfoque de minería de datos que se ajusta dinámicamente a los cambios en los patrones de fraude, permitiendo una detección más efectiva.
- Aplicación de tecnologías de Big Data: Se utiliza un sistema distribuido para procesar grandes volúmenes de datos, facilitando el análisis de más de 1 terabyte de registros de detalles de llamadas (CDR) de operadores mayoristas y transitarios de telecomunicaciones.
- Detección de fraude de participación en ingresos internacionales (IRS): La solución propuesta ha demostrado ser eficaz en la detección de fraudes IRS con menos del 5% de falsos positivos, utilizando datos históricos y en tiempo real.

Resultados

- Precisión en la detección: La solución mostró una tasa de falsos positivos inferior al 5%, lo que indica una alta precisión en la identificación de fraudes.
- Escalabilidad: La aplicación de tecnologías de Big Data permitió procesar más de 1 terabyte de datos, demostrando la capacidad del sistema para manejar grandes volúmenes de información.
- Eficiencia operativa: La implementación de la solución resultó en una detección más rápida y precisa de fraudes, mejorando la eficiencia operativa de los operadores de telecomunicaciones.

2.4 Planteamiento del problema

El avance de las tecnologías de comunicación ha transformado radicalmente la infraestructura y los servicios ofrecidos por los operadores de telecomunicaciones. Sin embargo, esta evolución también ha generado nuevas oportunidades para la aparición de fraudes técnicos cada vez más sofisticados. Estos fraudes no solo suponen una pérdida económica significativa para los operadores, sino que además comprometen la seguridad, calidad de servicio y confianza del usuario.

Según los últimos informes de la CFCA (Communications Fraud Control Association, 2023) y ENISA (2024), el impacto económico del fraude técnico en las telecomunicaciones representa pérdidas anuales superiores a 38 mil millones de dólares a nivel global, siendo Europa y en particular España zonas críticas por la elevada actividad fraudulenta vinculada a la interconexión internacional, portabilidad y roaming.

A pesar de la existencia de múltiples soluciones en el ámbito comercial, estas tecnologías presentan importantes limitaciones:

- Requieren reglas predefinidas o firmas estáticas que no detectan fraudes emergentes.

- Tienen una capacidad limitada de adaptación ante patrones dinámicos de comportamiento fraudulento.
- Pocas soluciones comerciales incorporan técnicas de aprendizaje automático no supervisado o grafos dinámicos que están siendo exploradas con éxito en el entorno académico.
- Existe escasa integración entre la investigación científica actual y los sistemas operativos reales, lo que genera una brecha entre la innovación y la implementación efectiva.

Por otro lado, los enfoques académicos —si bien han avanzado en el uso de machine learning, análisis semántico de llamadas o modelos de representación dinámica— sufren de una falta de validación en entornos de producción. La mayoría se basa en datasets limitados, no actualizados o simulados, lo que reduce su aplicabilidad real.

Esta disociación entre el estado del arte científico y la industria operativa deja sin resolver un problema clave:

La carencia de un marco técnico robusto, adaptable y validado para la detección automatizada de fraude técnico en telecomunicaciones que pueda aplicarse eficazmente en redes reales, especialmente en el contexto europeo y español, donde las amenazas continúan evolucionando más rápido que las capacidades de detección actuales.

En consecuencia, se identifica una necesidad urgente de:

- Integrar modelos de análisis de comportamiento, aprendizaje automático y detección en tiempo real con infraestructuras operativas.
- Desarrollar sistemas híbridos que combinen detección basada en reglas y detección inteligente adaptativa.
- Aplicar estas metodologías en un caso de estudio práctico para el contexto español y compararlo con otros entornos europeos.

Este proyecto se propone cerrar esta brecha crítica mediante la aplicación de metodologías actuales (como CRISP-DM), algoritmos de detección avanzados y la validación empírica sobre datos simulados.

Capítulo 3. OBJETIVOS

Este capítulo recoge los objetivos que guían el desarrollo del proyecto, comenzando por el objetivo general que define la meta principal a alcanzar. A continuación, se detallan los objetivos específicos, que desglosan en acciones concretas las tareas necesarias para cumplir con el propósito central. Finalmente, se presentan los beneficios esperados para la industria de las telecomunicaciones, resaltando el impacto positivo de la propuesta en términos de reducción de fraude, optimización de recursos y mejora del servicio para el consumidor.

3.1 Objetivos generales

Diseñar, desarrollar e implementar un sistema integral de detección de fraude en el ámbito de las telecomunicaciones, orientado a identificar, analizar y mitigar en tiempo real las actividades ilícitas o no autorizadas que afectan la integridad de los servicios ofrecidos por los operadores. El sistema debe apoyarse en tecnologías avanzadas como inteligencia artificial, aprendizaje automático, análisis de comportamiento y minería de datos, permitiendo detectar patrones anómalos que evidencien posibles fraudes como la suplantación de identidad, el uso indebido del servicio, el bypass de llamadas, entre otros.

Este objetivo busca no solo reducir significativamente las pérdidas económicas asociadas al fraude, sino también fortalecer los mecanismos de control interno, mejorar la experiencia del usuario final y garantizar el cumplimiento de normativas legales y regulatorias en materia de seguridad de la información. Además, se espera que el sistema proporcione información útil para la toma de decisiones estratégicas en materia de gestión de riesgos, prevención de delitos cibernéticos y optimización de procesos operativos dentro de la empresa.

La creciente digitalización de los servicios de telecomunicaciones ha transformado radicalmente la forma en que las personas y organizaciones acceden a la información y se comunican. Sin embargo, este avance también ha generado un incremento proporcional en las amenazas de fraude, que afectan tanto a los operadores como a los usuarios finales. Las pérdidas derivadas de estas actividades fraudulentas no solo representan un riesgo financiero significativo para las empresas del sector, sino que también comprometen la calidad del servicio, la confianza del cliente y el cumplimiento normativo en materia de seguridad y protección de datos.

En este contexto, el desarrollo de un sistema de detección de fraude se vuelve una necesidad estratégica. Este sistema permitiría a las compañías identificar, analizar y responder de forma proactiva a eventos sospechosos, antes de que generen consecuencias mayores. El objetivo general de este proyecto —centrado en el diseño e implementación de un sistema automatizado e inteligente de detección— responde directamente a esta necesidad.

Además, el uso de tecnologías como el aprendizaje automático y el análisis de grandes volúmenes de datos (big data) ofrece una ventaja significativa frente a los métodos tradicionales de monitoreo, que suelen ser estáticos, lentos y poco adaptables ante nuevas formas de fraude. Un sistema moderno y adaptativo no solo optimiza la eficiencia operativa y reduce los falsos

positivos, sino que también se convierte en una herramienta de apoyo para la toma de decisiones estratégicas y la planificación de políticas de seguridad más robustas.

Por tanto, el cumplimiento del objetivo general contribuirá de manera directa a:

- Reducir pérdidas económicas asociadas a actividades fraudulentas.
- Proteger la integridad de los sistemas de telecomunicaciones.
- Garantizar la continuidad y confiabilidad del servicio.
- Cumplir con normativas legales y regulatorias en materia de seguridad.
- Fortalecer la reputación y la confianza del cliente hacia la empresa.

En conclusión, este proyecto no solo tiene un valor técnico y operativo, sino también económico y estratégico, posicionándose como una solución necesaria frente a un problema creciente, complejo y altamente dinámico.

3.2 Objetivos específicos

Se va a utilizar la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), siendo ésta una de las más reconocidas y aplicadas en proyectos de minería de datos y analítica avanzada. Esta metodología es especialmente adecuada para problemas complejos como la detección de fraude técnico, ya que permite estructurar el trabajo desde el entendimiento del negocio hasta la puesta en producción.

1. Comprensión del Negocio (Business Understanding)

   Este objetivo busca realizar un estudio exhaustivo de las distintas modalidades de fraude que afectan a los operadores, tales como el *SIM boxing*, fraude por suplantación de identidad, acceso no autorizado, fraude de suscripción, interconexión fraudulenta y robo de servicio. El análisis incluirá su funcionamiento, impacto financiero, frecuencia y mecanismos de evasión utilizados por los defraudadores, a fin de establecer una base sólida para la construcción de modelos de detección eficaces.

   Actividades:
   - Entender el impacto del fraude técnico en la operadora (pérdidas económicas, reputación, carga de red).
   - Identificar y clasificar los tipos de fraude más comunes en el sector de las telecomunicaciones.
   - Identificación de indicadores clave de rendimiento (KPIs) para medir el éxito del sistema.

   Justificación:
   Permite enfocar el análisis en fraudes que realmente afectan los KPIs del negocio.

2. Comprensión de los Datos (Data Understanding)

Este objetivo tiene como propósito recopilar datos históricos y en tiempo real relacionados con llamadas, tráfico de datos, patrones de facturación, registros de cliente (KYC), uso de servicios y logs del sistema. El procesamiento y depuración de estos datos permitirá construir conjuntos de entrenamiento para los modelos de análisis y facilitará la identificación de patrones atípicos o sospechosos.

Actividades:

- Recopilar y procesar datos históricos de tráfico, facturación y comportamiento de usuarios para identificar patrones sospechosos.
- Examinar la calidad, formato, frecuencia, volumen y estructura de los datos.
- Identificar valores faltantes, duplicados, inconsistencias.
- Identificación de variables potencialmente predictivas del fraude.

Justificación:

El conocimiento profundo de los datos es vital para detectar patrones anómalos.

3. Preparación de los Datos (Data Preparation)

Transformar, limpiar, integrar y seleccionar los datos necesarios para construir conjuntos de entrenamiento y prueba que permitan alimentar modelos de detección de fraude con precisión y coherencia.

Actividades:

- Limpieza de datos: eliminación de ruido, corrección de errores, imputación de valores nulos.
- Generación de atributos (feature engineering): duración promedio de llamadas, número de BTS por día, cambio de IMEI, etc.
- Normalización y codificación de variables categóricas.

Justificación:
Aumenta la capacidad predictiva de los modelos y mejora la calidad del análisis.

4.Modelado (Modeling)

Mediante este objetivo se desarrollarán algoritmos capaces de detectar comportamientos anómalos asociados al fraude, basados en técnicas de minería de datos, inteligencia artificial y aprendizaje automático

Se evaluarán distintas metodologías para seleccionar la más adecuada según la naturaleza de los datos y la precisión deseada.

Modelos a aplicar:

- No supervisados: clustering (k-Means, Autoencoder, Isolation Forest) para detectar patrones anómalos.
- Supervisados: Random Forest, XGBoost, SVM si se cuenta con etiquetas de fraude conocidas.

Técnicas complementarias:

- Técnicas de reducción de dimensionalidad (PCA) para visualización.

Actividades:

- Selección de algoritmos apropiados según el tipo de datos y objetivos del negocio.
- Entrenamiento de los modelos con datos históricos etiquetados (cuando estén disponibles).
- Validación cruzada para mejorar la generalización del modelo.
- Ajuste de hiperparámetros para maximizar precisión, sensibilidad y especificidad.

Justificación:
Permite detectar tanto fraudes conocidos como nuevos comportamientos sospechosos.

5. Evaluación (Evaluation)

Aquí se medirá la efectividad del sistema utilizando métricas. Esta evaluación permitirá validar los resultados del sistema y definir oportunidades de mejora continua.

Actividades:

- Comparación entre distintos modelos para seleccionar el más adecuado.
- Validar los modelos con métricas: precisión, recall, F1-score, AUC.
- Simulación con casos artificiales de fraude.
- Evaluación de impacto operacional y tasa de falsos positivos.

Justificación:
Asegura que los modelos sean útiles en un entorno real y confiables para decisiones automáticas.

6. Despliegue (Deployment)

Implementar el sistema de detección de fraude en el entorno productivo de la empresa, integrándolo con las plataformas existentes de monitoreo, alertas y gestión de incidencias, y garantizando su funcionamiento continuo y escalable.

Actividades:

- Integración con sistemas OSS/BSS.
- Creación de un dashboard para alertas y visualización de eventos.
- Automatización del flujo de datos y generación de reportes en tiempo real.
- Capacitación al personal encargado de la supervisión y respuesta ante incidentes.
- Establecimiento de un plan de mantenimiento y actualización del sistema.

Justificación:
Transforma un modelo analítico en una solución funcional que protege la red en tiempo real.

3.3 Beneficios del proyecto

El proyecto aporta beneficios significativos en distintos niveles, que van desde el impacto económico directo hasta la contribución a la seguridad y la escalabilidad de la solución. Estos beneficios no solo fortalecen a los operadores de telecomunicaciones frente al fraude, sino que también repercuten en los usuarios, el entorno regulatorio y la industria en general.

1. Beneficios Económicos

- Reducción de pérdidas por fraude: La detección temprana de fraude técnico (como SIM Box, suplantación de identidad o desvío de tráfico) puede reducir hasta un 30-50% de las pérdidas operativas derivadas de llamadas no facturadas o tráfico enrutado ilegalmente.
- Mejor recuperación de ingresos (Revenue Assurance): Al detectar manipulaciones de CDRs o tráfico en tiempo real, se recuperan ingresos que antes quedaban fuera del sistema de facturación.
- Reducción de sanciones regulatorias: Evitar el uso indebido de recursos asignados (como IMSIs) y mejorar el cumplimiento normativo.

2. Beneficios Tecnológicos

- Desarrollo de un sistema de detección inteligente adaptativo: El uso de modelos de Machine Learning y análisis no supervisado permite descubrir fraudes nuevos (zero-day) sin necesidad de reglas predefinidas.
- Arquitectura escalable y reutilizable: La solución podrá integrarse a futuro con sistemas OSS/BSS de cualquier operador, y escalarse a entornos 5G, VoLTE o IoT.

3. Beneficios Operativos

- Optimización de recursos humanos: El sistema reduce la carga del personal técnico al automatizar procesos de revisión y escaneo de datos masivos.
- Tiempo de reacción más rápido: Gracias a alertas en tiempo real, los operadores pueden tomar decisiones inmediatas (bloqueo, seguimiento, denuncia).
- Detección proactiva vs. reactiva: Se pasa de un enfoque reactivo a uno predictivo, anticipando fraudes antes de que escalen.

4. Beneficios Sociales y Regulatorios

- Mejora de la calidad del servicio: Al reducir el tráfico fraudulento, se mejora la disponibilidad y el rendimiento real para los usuarios legítimos.
- Protección del consumidor: Se evita que usuarios sean víctimas de suplantación de identidad, cobros indebidos o manipulación de datos móviles.
- Contribución a la seguridad nacional y ciberseguridad: La detección de fraudes en infraestructura crítica fortalece la resiliencia frente a amenazas internas y externas.

5. Beneficios para la Escalabilidad del Proyecto

- Posibilidad de replicación en otros operadores: El modelo puede adaptarse fácilmente a otros contextos nacionales o internacionales.

Capítulo 4. DESARROLLO DEL PROYECTO

Este capítulo se centra en el desarrollo del proyecto, detallando las fases, decisiones y recursos que permitieron llevar a cabo la propuesta de detección y clasificación de fraude en telecomunicaciones. En primer lugar, se describe la planificación del proyecto, estableciendo la metodología de investigación y los hitos definidos. Posteriormente, se presenta la solución diseñada, especificando su arquitectura, las metodologías y herramientas empleadas, así como el pipeline de procesamiento de datos y los algoritmos seleccionados.

Asimismo, se exponen los recursos requeridos tanto tecnológicos como humanos, junto con el presupuesto estimado para el desarrollo, implementación y mantenimiento del sistema. El capítulo también analiza la viabilidad técnica, económica y operativa de la propuesta. Finalmente, se presentan los resultados obtenidos a partir de la experimentación y validación, destacando el desempeño de los algoritmos seleccionados y la efectividad del sistema en la detección de fraudes.

4.1 Planificación del proyecto

La planificación del proyecto se ha diseñado siguiendo la metodología CRISP-DM (Cross Industry Standard Process for Data Mining), reconocida como un marco de referencia en proyectos de analítica avanzada. Esta metodología permite estructurar el trabajo en seis fases iterativas, garantizando un desarrollo ordenado y coherente que va desde la comprensión inicial del problema hasta el despliegue de la solución. A continuación, se describen las actividades realizadas en cada fase, junto con su justificación y un cronograma estimado de ejecución en semanas.

Fase 1: Comprensión del Negocio

En esta etapa se llevó a cabo una revisión bibliográfica y documental sobre las principales modalidades de fraude técnico en telecomunicaciones, tales como *SIM boxing*, suplantación de identidad e interconexión fraudulenta. Se analizaron informes técnicos y regulatorios del sector para identificar y clasificar los distintos tipos de fraude, lo que permitió establecer los KPIs relevantes para medir el impacto de estas prácticas, incluyendo pérdidas económicas, congestión de red y tasa de cancelación de servicios.

Fase 2: Comprensión de los Datos

Se procedió con la creación de un dataset sintético, seguido de un análisis preliminar de calidad de datos para detectar valores nulos, duplicados e incoherencias. Adicionalmente, se desarrolló un análisis exploratorio de datos (*EDA*) mediante técnicas de visualización y estadísticas descriptivas, lo que facilitó una primera aproximación a patrones de comportamiento relacionados con posibles fraudes.

Fase 3: Preparación de los Datos

En esta fase se ejecutaron tareas de limpieza y transformación de datos, incluyendo la corrección de errores y la imputación de valores faltantes. Se generaron atributos derivados de

especial relevancia, como llamadas internacionales, destinos premium o llamadas en horarios inusuales (madrugada). También se aplicaron procesos de normalización y codificación de variables categóricas, asegurando la consistencia de los datos para su posterior modelado. Todo este proceso fue documentado rigurosamente para garantizar la reproducibilidad y transparencia en la construcción del *dataset* final.

Fase 4: Modelado

Se realizaron pruebas con algoritmos supervisados (Random Forest y XGBoost) aplicados sobre conjuntos de datos previamente etiquetados, complementadas con modelos no supervisados como *k-Means* e *Isolation Forest* para la detección de anomalías. Se emplearon ajuste de hiperparámetros para optimizar el rendimiento de los modelos. Cada experimento fue documentado y se elaboró un registro comparativo de resultados que permitió seleccionar las configuraciones más prometedoras.

Fase 5: Evaluación

Los modelos obtenidos fueron evaluados en función de métricas como precisión, *recall*, F1-score y AUC. Se prestó especial atención al análisis de la tasa de falsos positivos, buscando un equilibrio entre sensibilidad y especificidad. Finalmente, se revisó el impacto operativo y se efectuó una validación funcional para asegurar la aplicabilidad del sistema en entornos reales.

Fase 6: Despliegue

Aunque esta fase corresponde a actividades futuras, se ha definido un plan de acción que contempla: el diseño y despliegue de un sistema de alertas y visualización, la automatización del flujo de entrada de datos y ejecución de modelos, la integración con sistemas OSS/BSS de la operadora, la capacitación básica del personal de operación y el establecimiento de políticas de mantenimiento y actualización del sistema.

Cronograma:

Actividades/ Tareas	Semana 1 - 2	Semana 3 - 4	Semana 5	Semana 6 - 7 - 8	Semana 9 - 10 - 11	Semana 12 - 13
Fase 1: Comprensión del Negocio						
Fase 2: Comprensión de los Datos						
Fase 3: Preparación de los Datos						

Fase 4: Modelado	
Fase 5: Evaluación	
Memoria Final	

Tabla 4 - Cronograma

4.2 Descripción de la solución, metodologías y herramientas empleadas

El presente proyecto aborda la detección automática de fraude técnico en telecomunicaciones con análisis de registros de llamadas (Call Detail Records o CDR). Dada la sensibilidad de los datos reales en este ámbito, se ha optado por la generación de un dataset sintético controlado, que simula comportamientos normales y fraudulentos inspirados en escenarios reales como: SIM box, Bypass Fraud, International Revenue Share Fraud (IRSF), Wangiri y CLI Spoofing.

4.2.1 Descripción del dataset

Se ha generado un dataset sintético con las siguientes características:

- Registros simulados: 100.000
- Etiqueta de fraude: atributo binario donde “0” es legítimo y “1”es fraudulento. La proporción de fraude es del 30%.
- Campos incluidos:

Nombre de la variable	Tipo	Descripción
usuario_id	int64	Identificador de usuario
duracion_seg	float64	Duración de la llamada en segundos
numero_destino	int64	Número de teléfono de destino
pais_origen	object	País desde donde se origina la llamada
pais_destino	object	País destino de la llamada
hora_llamada	datetime64[ns]	Fecha y hora de la llamada
tipo_llamada	object	Tipo de llamada: Local, Internacional o roaming
costo_usd	float64	Coste de la llamada en dólares
es_fraude	int64	Etiqueta de fraude

Tabla 5 - Campos del Dataset

- No hay valores faltantes ni tampoco duplicados.
- La distribución de fraude es:
    - es_fraude = 1 (fraudulento) -- > 74,93%
    - es_fraude = 0 (legitimo) -- > 25,07 %

4.2.2 Feature Engineering

El objetivo del feature engineering es mejorar la separabilidad entre patrones legítimos y fraudulentos y capturar aspectos conductuales como horarios atípicos, costos extremos y desajustes entre localización y origen de llamada.

Para esto se generaron nuevas variables predictivas:

Nombre de la variable	Tipo	Descripción
hora	int32	Extraída de hora_llamada
dia_semana	int23	Extraída de hora_llamada
es_madrugada	int64	Llamada realizada entre 00:00 y 06:00
llamada_internacional	int64	Si el país origen es distinto al país de destino
destino_premium	int64	Si el numero de destino empieza por prefijos 900/899

Tabla 6 - Variables Predictivas Generadas

En un segundo paso se hizo una codificación de las variables categóricas mediante LabelEncoder.

Y por último, como tenemos una distribución del fraude desbalanceado se utilizó el RamdomUnderSampler() para tener un peso equitativo en las muestras.

Una vez tenemos los datos preparados, los separamos en entrenamiento y testing y estandarizamos los datos con Standardscaler, transformando cada columna numérica para que tenga media 0 y desviación estándar 1.

Esta estandarización es importante ya que muchos de los modelos lineales son sensibles a la escala de los datos. Si una variable tiene valores grandes y otra valores pequeños, la primera dominara el modelo y esto puede llevar a coeficientes mal ajustados o/y una convergencia lenta o fallida del optimizador.

El escalado es importante en modelos que optimizan funciones sensibles a la magnitud, como LogisticRegression, y los que se basan en distancias, como SVC, KNN y KMeans, pero sin embargo no es necesario para arboles de decisión y RandomForest, ya que estos no son sensibles al escalado de los datos.

4.2.3 Modelado

Principios de elección de métodos:

1. Naturaleza del dato y disponibilidad de etiquetas

En muchos escenarios de fraude, no se dispone de suficientes datos etiquetados con instancias de fraude confirmadas (problema de clases desbalanceadas).

Por tanto:

- Se emplean métodos no supervisados (clustering, detección de anomalías) para descubrir patrones sospechosos sin etiquetas.
- Cuando hay etiquetas fiables, se aplican métodos supervisados (árboles de decisión, SVM) para maximizar la precisión de detección.

2. Capacidad para detectar fraudes desconocidos o emergentes

- El fraude técnico es dinámico: aparecen variantes constantemente.
- Los métodos no supervisados permiten detectar desviaciones del comportamiento normal (lo que puede indicar fraude nuevo).
- Los supervisados pueden volverse obsoletos si solo reconocen patrones ya conocidos.

3. Interpretabilidad vs. precisión

- Métodos como random forest o XGBoost ofrecen buen rendimiento y cierta interpretabilidad.
- Técnicas como autoencoders o deep learning pueden ser más precisas, pero menos explicables.
- En entornos críticos como las telecomunicaciones, se valora que el modelo justifique las alertas para su validación.

4. Escalabilidad y rendimiento en tiempo real

- El tráfico en telcos es masivo y continuo.
- Se prefieren modelos ligeros o preentrenados, como:
    - Supervisados rápidos: árboles, regresión logística.
    - No supervisados escalables: k-means, Isolation Forest.

5. Coste de error (falsos positivos vs. falsos negativos)

- En fraude, los falsos negativos (fraudes no detectados) suelen ser más costosos que los falsos positivos (alarmas falsas).
- Métodos supervisados permiten ajustar umbrales y priorizar la sensibilidad del modelo.
- Métodos no supervisados ayudan a refinar alertas para reducir ruido.

6. Combinación híbrida como enfoque ideal

- Muchos sistemas de detección de fraude modernos utilizan enfoques híbridos:
    - Análisis no supervisado para descubrir anomalías y generar etiquetas.
    - Entrenamiento supervisado con esas etiquetas para clasificar futuros eventos.
- Ejemplo: usar clustering para identificar tráfico atípico y luego entrenar un clasificador supervisado con los casos validados.

Para el análisis de detección de fraude vamos a utilizar y comparar varios modelos:

- Modelos supervisados:
  El presente estudio se enmarca dentro de los modelos de aprendizaje supervisado, dado que se trata de un problema de clasificación, en el que cada instancia del dataset cuenta con una etiqueta conocida, denominada “es_fraude”, que indica si una llamada es fraudulenta o no. Este enfoque permite que los modelos aprendan a identificar patrones y relaciones entre las variables de entrada y la etiqueta objetivo, con el fin de predecir correctamente nuevas observaciones.

  En este contexto, se comparan distintos modelos supervisados para determinar cuál ofrece el mejor desempeño en la detección de fraude.

  Modelos usados:

Modelo	Tipo de Modelo	Motivo de selección
XGBoost	Ensamble - Boosting	Alto rendimiento, robusto e interpretable
Random Forest	Ensamble - Bagging	Modelo base, ideal para relaciones no lineales
Logistic Regression	Lineal	Baseline interpretable
SVC	Kernel-based	Útil para detectar fronteras no lineales

Tabla 7 - Modelos Supervisados Usados

XGBoost:

XGBoost es un algoritmo de aprendizaje supervisado basado en boosting, que construye modelos predictivos mediante la combinación secuencial de árboles de decisión. Cada nuevo árbol se entrena para corregir los errores residuales del modelo

anterior, utilizando técnicas de gradiente descendente para optimizar la función de pérdida. Esta aproximación permite que el modelo reduzca de manera eficiente el error de predicción y mejore su rendimiento en comparación con métodos individuales como un único árbol de decisión o incluso Random Forest.

Ventajas	Desventajas
Altísimo rendimiento en muchos datasets reales. Muy eficiente y optimizado (maneja nulos, early stopping, etc). Flexible: soporta distintos tipos de funciones de pérdida y regularización, lo que ayuda a prevenir overfitting.	Mas complejo de configurar. Mas difícil de interpretar que el Random Forest. Mayor demanda computacional.

Tabla 8 – XGBoost Ventajas/Desventajas

Random Forest:

Random Forest es un algoritmo de aprendizaje supervisado basado en bagging que crea múltiples árboles de decisión a partir de diferentes subconjuntos del dataset y combina sus predicciones promediando los resultados (para problemas de regresión) o mediante votación mayoritaria (para clasificación). Cada árbol se construye a partir de una muestra aleatoria del conjunto de datos, y en cada nodo se selecciona un subconjunto aleatorio de características para la división, lo que introduce diversidad entre los árboles y reduce la varianza del modelo.

Ventajas	Desventajas
Robusto al overfitting. Maneja variables categóricas y numéricas. Detecta relaciones no lineales. Relativamente fácil de entrenar y menos sensible a valores atípicos que un único árbol de decisión.	Difícil de interpretar. Puede ser más lento en predicción que modelos simples. Puede requerir más memoria y capacidad computacional que modelos más sencillos.

Tabla 9 - Random Forest Ventajas/Desventajas

Logistic Regression:

Regresión logística es un modelo de aprendizaje supervisado utilizado principalmente para problemas de clasificación binaria. Su objetivo es ajustar una función logística (sigmoide) que modela la probabilidad de que una observación

pertenezca a una clase específica (0 o 1). La salida del modelo es un valor entre 0 y 1, que puede interpretarse como la probabilidad de pertenencia a la clase positiva.

Ventajas	Desventajas
Muy rápido y fácil de interpretar. Buena base para establecer un benchmark. Funciona bien si los datos son linealmente separables. Requiere relativamente pocos recursos computacionales.	No captura relaciones no lineales. Menos potente en problemas complejos. Sensible a variables altamente correlacionadas y a datos no balanceados, lo que puede requerir preprocesamiento adicional.

Tabla 10 - Logistic Regression Ventajas/Desventajas

SVC:

El Support Vector Classifier (SVC) es un modelo de aprendizaje supervisado utilizado para problemas de clasificación. Su objetivo principal es encontrar un hiperplano óptimo que separe las clases, maximizando el margen entre las observaciones de diferentes categorías. Para problemas donde las clases no son linealmente separables, SVC permite el uso de kernels (como RBF, polinómico o sigmoide) que transforman el espacio de características y capturan relaciones no lineales entre las variables.

Ventajas	Desventajas
Preciso en espacios de alta dimensión. Capaz de capturar no linealidades con kernels como rbf. Robusto a problemas con margen estrecho entre clases.	Escala mal con datasets grandes. Mas difícil de interpretar. Requiere estandarización.

Tabla 11 - SVC Ventajas/Desventajas

- Modelos No Supervisados:

Para complementar la detección de fraude, se emplean modelos de aprendizaje no supervisado, cuyo objetivo es identificar nuevos patrones y anomalías que podrían indicar actividades fraudulentas. Esta elección se justifica por varias razones:

  - El fraude técnico es dinámico: aparecen variantes constantemente.
  - Los métodos no supervisados permiten detectar desviaciones del comportamiento normal (lo que puede indicar fraude nuevo).

- Los métodos supervisados pueden volverse obsoletos si solo reconocen patrones ya conocidos.

En este estudio, se utilizará inicialmente clustering para identificar grupos y categorías de fraude existentes, seguido de técnicas como autoencoders e Isolation Forest para la detección de anomalías. Estas técnicas permiten identificar valores atípicos que no encajan con los patrones previamente clasificados, lo que proporciona un enfoque más robusto y adaptable frente a fraudes emergentes.

Modelos usados:

Modelo	Motivo de selección
Clustering (K-means)	Para dividir los fraudes en grupos
Autoencoders	Para detectar anomalías
Isolation Forest	Para detectar anomalías
Clustering (K-means)	Para dividir los fraudes en grupos

Tabla 12 - Modelos No Supervisados Usados

Clustering – K-Means:

K-Means es un algoritmo de aprendizaje no supervisado que busca agrupar un conjunto de puntos en k clústeres, de manera que la distancia interna dentro de cada grupo sea mínima. Generalmente se utiliza la distancia euclidiana al cuadrado como medida de similitud, y cada clúster se representa mediante su centroide, que se recalcula iterativamente hasta que las asignaciones de los puntos a los clústeres se estabilizan.

Ventajas	Desventajas
Muy rápido y simple.	Requiere definir k de antemano.
Buena base para segmentación.	Sensible a escala y outliers.
Flexible para distintos tipos de aplicaciones.	Solo detecta formas esféricas (no clústeres irregulares).

Tabla 13 - K-Means Ventajas/Desventajas

Autoencoders:

Los autoencoders son un tipo de red neuronal no supervisada que se entrenan para comprimir (encoder) y luego reconstruir (decoder) los datos de entrada. Durante el entrenamiento, la red aprende a capturar las características más

relevantes de los datos mientras minimiza el error de reconstrucción. Este error se utiliza como indicador de anomalía, ya que los datos que difieren significativamente de los patrones aprendidos generan un mayor error de reconstrucción, lo que permite identificar posibles fraudes o comportamientos inusuales.

Ventajas	Desventajas
Captura relaciones no lineales complejas. Útil para reducción de dimensionalidad o detección de anomalías. Flexible y adaptable a diferentes tipos de datos, incluyendo secuencias temporales y datos multivariables	Requiere buen diseño y entrenamiento (no trivial). Difícil de interpretar. Demanda recursos computacionales importantes para datasets grandes.

Tabla 14 - Autoencoders Ventajas/Desventajas

Isolation Forest:

Isolation Forest es un algoritmo de aprendizaje no supervisado diseñado para la detección de anomalías. Su principio básico es que las observaciones atípicas (anómalas) son más fáciles de aislar que las normales. Para ello, el modelo construye múltiples árboles aleatorios, seleccionando características y valores de corte al azar, de manera que las anomalías queden separadas en menos pasos que las observaciones normales.

Ventajas	Desventajas
Rápido, incluso en datasets grandes. Funciona bien en alta dimensión. No requiere escalado de carácterísticas.	Menos efectivo si las anomalías no están bien separadas. No captura relaciones complejas como un autoencoder. Puede requerir ajuste cuidadoso del número de árboles y submuestras para optimizar el rendimiento.

Tabla 15 - Isolation Forest Ventajas/Desventajas

Estrategia: *segmentar el fraude según sus características*

Tras aplicar técnicas de clustering para agrupar los fraudes según sus características, el siguiente paso consiste en analizar cada grupo para determinar si corresponde a patrones de fraude previamente identificados o a nuevas variantes. Esta segmentación permite entender mejor la estructura del fraude, identificar tendencias emergentes y diseñar estrategias de detección más precisas. Además,

facilita la priorización de esfuerzos de mitigación, ya que se pueden asignar recursos y controles específicos a cada tipo de fraude detectado, optimizando la efectividad del sistema de prevención.

Patrones típicos de fraude:

Posible tipo de fraude	Características esperadas
Wangiri	Duración muy corta (1 seg), costo bajo
IRSF (premium)	Número destino premium, larga duración, alto costo
SIM box / Bypass	Llamadas locales con destinos internacionales
CLI Spoofing	Número origen no consistente con país origen
General anomalías	% llamadas de madrugada, llamadas frecuentes, roaming

Tabla 16 - Patribes típicos de Fraude

Con clustering obtenemos:

- Grupos de fraude con patrones distintos
- Posible correspondencia con tipos conocidos (aunque no etiquetados)
- Insight útil para crear reglas o alimentar modelos futuros

4.2.4 Evaluación

Detección del Fraude – Modelos Supervisados

La evaluación se diseñó para comparar el desempeño de diferentes algoritmos de clasificación para determinar cuál ofrece el mejor rendimiento.

Para evaluar de forma objetiva cada modelo, se utilizaron las siguientes métricas de clasificación binaria:

Métrica	Descripción		
Confussion Matrix	Matriz 2x2 que compara las predicciones del modelo contra los valores reales:		
		Predicho No Fraude (0)	*Predicho Fraude (1)*
	Real No Fraude (0)	**TN** (True Negative)	**TN** (True Negative)
	Real Fraude (1)	**N** (False Negative)	**N** (False Negative)

Classification Accuracy	Con que frecuencia acierta el clasificador
Classification Error / Missclassification Rate	Con que frecuencia el clasificador se equivoca
Sensitivity / True Positive Rate / Recall	Cuando el valor actual es positivo, con que frecuencia el clasificador acierta
Specifity	Cuando el valor actual es negativo, con qué frecuencia el clasificador acierta
False Positive Rate	Cuando el valor actual es negativo, con qué frecuencia el clasificador se equivoca
Precision	Cuando se predice un valor positivo, con que frecuencia el clasificador acierta.
F1-score	Combina precisión y recall para dar una medida balanceada.
AUC-ROC	Area bajo la curva ROC Mide la capacidad del modelo de distinguir entre fraude y no fraude en múltiples umbrales

Tabla 17 - Métricas Usadas

En un problema de fraude, como es este caso, lo importante es:

- Reducir los FN (Falsos Negativos), es decir, los fraudes reales que no el modelo no detecto.
  En otras palabras, necesitamos buscar un Sensitivity/Recall alto.
- Controlar los FP (Falsos Positivos), es decir, los valores que son detectados como fraude y realmente no los son.
  Alta tasa de falsos positivos puede generar alertas innecesarias, afectando la experiencia de usuario y carga de revisión manual.
- Buscar un balance entre Sensitivity y Precision.
  Es mejor clasificar un valor como fraudulento, aunque este no lo fuera, que no detectar la situación de fraude.

Proceso de evaluación del modelo:

1. Obtenemos el score del modelo tanto para entrenamiento como para el conjunto de test.
2. Imprimimos la Confussion Matrix y obtenemos todas las métricas posibles provenientes de ella:

a. Classification Accuracy – es la métrica más fácil de entender, pero no nos dice qué tipo de errores se están cometiendo.
b. Classification Error
c. Sensitivity/Recall
d. Specificity
e. False Positive Rate
f. Precision

Con todas estas métricas y el problema de fraude actual podemos decidir cuáles de ellas utilizar.

Para evaluar un problema de fraude es necesario optimizar la Sensitivity.

3. Una vez tenemos esto, imprimimos los 25 primeros valores "verdaderos" y las 25 primeras predicciones. Las comparamos y vemos que tipo de errores está cometiendo el modelo.
   La mayoría de los modelos cometen fallos al predecir el fraude.
4. Obtenemos el Classification Report para tener en una vista rápida un resumen de las métricas de precisión, recall y f1-score.
5. Ajustamos el modelo para optimizar la Sensitivity pero teniendo un equilibrio con la precisión y el f1-score.
   Dibujamos esta gráfica para poder ver de manera más sencilla el valor elegido.
6. Evaluamos con ROC Curves y Area Under the Curve (AUC), la calidad general del modelo.
7. Por último, usamos la validación cruzada para obtener una estimación más robusta de la media, recall y la desviación estándar.

Este análisis se realizó comparando las predicciones del modelo sobre los datos simulados con comportamientos conocidos de fraude, identificando si el sistema fallaba en detectar o generaba demasiadas falsas alarmas en situaciones inofensivas (como llamadas nocturnas no fraudulentas).

4.3 Recursos requeridos

En este apartado se enumeran los recursos técnicos y materiales empleados para la ejecución del proyecto de análisis de fraude técnico en el sector de telecomunicaciones. A continuación, se listan los equipos, herramientas de software y entornos de trabajo utilizados durante el desarrollo y validación de las pruebas.

- Equipo informático: Raspberry Pi 5, procesador ARM Cortex-A76 (quad-core, 2.4 GHz), 8 GB de memoria RAM.
- Lenguaje de programación: Python 3.
- Entorno de desarrollo: Jupyter Notebook.

4.4 Presupuesto

A continuación, se detalla la evaluación económica total del proyecto, incluyendo tanto recursos materiales como el tiempo de trabajo invertido. Se consideran valores de mercado aproximados para los equipos y software utilizados, aunque no se haya producido una compra directa. Esto permite valorar de forma realista el coste del proyecto si tuviera que ser replicado o escalado.

Tipo de coste	Valor	Comentarios
Horas de trabajo en el proyecto	240 horas	Incluye desarrollo, documentación y análisis. 240h x 30€ = 7.200 €
Equipo técnico utilizado	1.500 €	Desktop con 32 GB RAM, CPU 8 núcleos.
Software utilizado	0 €	Todo el software empleado ha sido libre o de uso gratuito, lo que ha permitido minimizar costes sin comprometer capacidad analítica.
Estudios e informes	0 €	Se ha trabajado con artículos e informes públicos.
Materiales empleados	0 €	Dataset sintético
TOTAL	8.700 €	

Tabla 18 - Presupuesto

4.5 Viabilidad

El presente apartado evalúa la viabilidad del sistema de detección de fraude técnico en telecomunicaciones desarrollado, analizando su factibilidad económica, sostenibilidad y riesgos asociados. Este análisis integral permite valorar la aplicabilidad real de la solución en entornos productivos de operadores de telecomunicaciones.

Viabilidad Económica

Contexto del Mercado y Magnitud del Problema

El fraude en telecomunicaciones representa una amenaza significativa para la industria global. Según el informe Global Fraud Loss Survey 2023 de la Communications Fraud Control Association (CFCA), las pérdidas por fraude representaron el 2,5% de los ingresos globales de telecomunicaciones, equivalentes a $38.95 mil millones, con un incremento del 12% respecto a 2021.

Sobre esta base, se propone una distribución prudente del impacto de las principales modalidades de fraude, tomando como referencia su relevancia relativa según los informes de la CFCA:

Tipo de Fraude	Impacto sobre Facturación
Bypass internacional y desvío de tráfico	0,3% – 0,8%
SIM swapping y clonación	0,1% – 0,4%
Servicios premium no autorizados	0,2% – 0,6%
Roaming fraudulento	0,1% – 0,3%

Tabla 19 - Fraude & Impacto sobre Fracturación

Análisis del Mercado Español

En 2023, las principales operadoras españolas (Telefónica, Vodafone, Orange y MásMóvil) generaron conjuntamente 25.178 millones de euros de ingresos. Aplicando el porcentaje de fraude del 2,5% identificado por CFCA, las pérdidas potenciales ascienden a aproximadamente 630 millones de euros anuales en el mercado español.

Coste del Proyecto Desarrollado

El sistema propuesto ha sido desarrollado con un coste total estimado de 8.700 €, que incluye:

- Tiempo de desarrollo e investigación.
- Recursos computacionales para entrenamiento de los modelos.
- Herramientas de software y licencias.

Análisis de Retorno de Inversión (ROI)

Para el cálculo del ROI, se adopta un escenario conservador basado en los siguientes supuestos:

Supuestos del modelo:

- Operadora con pérdidas anuales por fraude: 100 millones de euros
- Reducción de fraude detectable alcanzada: 1% del total
- Ahorro anual estimado: 1 millón de euros
- Inversión inicial: 8.700 euros

El ROI se calcula como:

$$ROI = \frac{(\text{Beneficio Neto} - \text{Inversión Inicial})}{\text{Inversión Inicial}} x\ 100$$

Sustituyendo valores:

$$ROI = \frac{(1.000.000€ - 8.700€)}{8.700€} x\ 100\ \approx 11.390\%$$

Este resultado demuestra una rentabilidad excepcional, con un periodo de recuperación de la inversión inferior a cuatro días operativos.

Análisis de Sensibilidad

Escenario	Pérdidas Anuales (M€)	Reducción Fraude	Ahorro Anual (€)	ROI
Pesimista	50	0,5%	250.000	2.774%
Base	100	1,0%	1.000.000	11.390%
Optimista	200	2,0%	4.000.000	45.875%

Tabla 20 - Análisis de Sensibilidad

Incluso en el escenario más pesimista, el ROI supera el 2.700%, confirmando la viabilidad económica robusta del proyecto.

Sostenibilidad a Futuro

La sostenibilidad del sistema es un aspecto clave para garantizar que la inversión realizada no solo sea rentable a corto plazo, sino que también mantenga su efectividad en el tiempo. En este sentido, resulta fundamental evaluar cómo el modelo puede adaptarse a nuevas formas de fraude, mantenerse actualizado con datos recientes y seguir operando con costes reducidos. Estos factores determinan su capacidad de integrarse de manera estable en los entornos de producción y de contribuir de forma continua a la seguridad de las telecomunicaciones.

- Escalabilidad: El sistema puede adaptarse fácilmente a otras operadoras o contextos geográficos, incorporando nuevos patrones de fraude.
- Actualización del modelo: Puede ser reentrenado periódicamente con nuevos datos etiquetados, lo que asegura su eficacia a largo plazo.
- Costes operativos bajos: El uso de herramientas de código abierto y ejecución en infraestructuras ligeras permite mantener costes reducidos en producción.
- Impacto medioambiental bajo: El proyecto no requiere componentes físicos ni equipamiento especializado, lo que reduce su huella ecológica.

Análisis de Riesgos

De forma complementaria, es necesario considerar los riesgos asociados a la implementación del sistema. Toda solución tecnológica, por robusta que sea, está expuesta a posibles limitaciones técnicas y operativas que pueden afectar su funcionamiento o su adopción dentro de una organización. Identificar estas amenazas desde el inicio permite diseñar estrategias de

mitigación adecuadas, reduciendo su impacto potencial y asegurando la continuidad del proyecto en el largo plazo.

Riesgos Técnicos

Riesgo	Probabilidad	Impacto	Mitigación
Degradación del modelo	Media	Alto	Monitorización continua y reentrenamiento
Problemas de integración	Baja	Medio	Pruebas exhaustivas y desarrollo incremental
Incompatibilidades de datos	Media	Medio	Análisis previo de formatos y ETL robusto

Tabla 21 - Riesgos Técnicos

Riesgos Operativos

Riesgo	Probabilidad	Impacto	Mitigación
Resistencia al cambio	Media	Medio	Programa de capacitación y demostración de valor
Falta de datos de calidad	Baja	Alto	Validación de calidad de datos y procesos de limpieza
Sobrecarga operativa	Baja	Medio	Automatización y interfaces intuitivos

Tabla 22 - Riesgos Operativos

Conclusión

El análisis realizado confirma que el proyecto es económicamente viable, con un retorno de la inversión (ROI) extraordinariamente positivo desde el primer año y un coste de desarrollo muy reducido frente al ahorro potencial que genera. Además, su diseño flexible y escalable asegura la sostenibilidad técnica y económica en el tiempo, permitiendo que el sistema se adapte a nuevas tipologías de fraude y mantenga bajos costes operativos gracias al uso de herramientas abiertas y procesos de actualización periódica.

No obstante, la implantación también conlleva riesgos técnicos y operativos que deben ser considerados, como la posible degradación del modelo, problemas de integración o la resistencia al cambio dentro de las organizaciones. La identificación temprana de estos riesgos y la definición de estrategias de mitigación son elementos clave para garantizar la continuidad y estabilidad del proyecto. En conjunto, la combinación de una alta rentabilidad, sostenibilidad a largo plazo y un plan de gestión de riesgos bien definido refuerza la viabilidad integral de la solución propuesta para la detección de fraude en telecomunicaciones.

4.6 Resultados del proyecto

En este apartado se presentan los resultados obtenidos tras el desarrollo e implementación del sistema de detección de fraude en telecomunicaciones. El objetivo principal fue construir una solución capaz de identificar ataques como SIM box, IRSF, Wangiri y CLI Spoofing. Para ello, se generó un dataset sintético realista, sobre el cual se aplicaron distintas técnicas de aprendizaje automático y se entrenaron múltiples modelos.

El análisis se centra en la evaluación comparativa de los enfoques supervisados y no supervisados, valorando su rendimiento, capacidad de detección de patrones fraudulentos y efectividad en escenarios de referencia. Asimismo, se examinan aspectos clave como la precisión, la interpretabilidad y la eficiencia operativa, con el fin de determinar qué modelos ofrecen el mejor equilibrio entre robustez técnica y aplicabilidad práctica. Los resultados obtenidos constituyen la base para medir el impacto real del proyecto y para valorar su potencial integración en entornos productivos de telecomunicaciones.

4.6.1 Evaluación comparativa de modelos supervisados para la detección de fraude

La siguiente tabla resume los resultados de precisión, estabilidad (desviación estándar) y F1-score obtenidos en validación cruzada:

Modelo	Precision	F1-score	Sensitivity	Desviación Estándar*	ROC_AUC
Random Forest	0.851	0.861	0.9372	0.0051	0.9553
Logistic Regression	0.6948	0.740	0.8783	0.0047	0.8022
SVC (RBF)	0.8081	0.821	0.8798	0.0040	0.9126
XGBoost	**0.8207**	**0.841**	**0.9796**	**0.0024**	**0.9558**

Tabla 23 - Comparativa Modelos Supervisados

*Desviación estándar del Recall/Sensitivity

El modelo **XGBoost** fue el seleccionado por las siguientes razones:

- Tiene la mejor sensitivity (detecta más fraudes).
- Tiene F1 alto, lo que equilibra con falsos positivos.
- Tiene muy buena estabilidad y ROC AUC.
- Y aunque Random Forest tiene un poco más de precisión, para fraude lo más importante es no dejar escapar fraudes (recall/sensitivity).

Gráfico Precision, Recall y F1-score vs Threshold

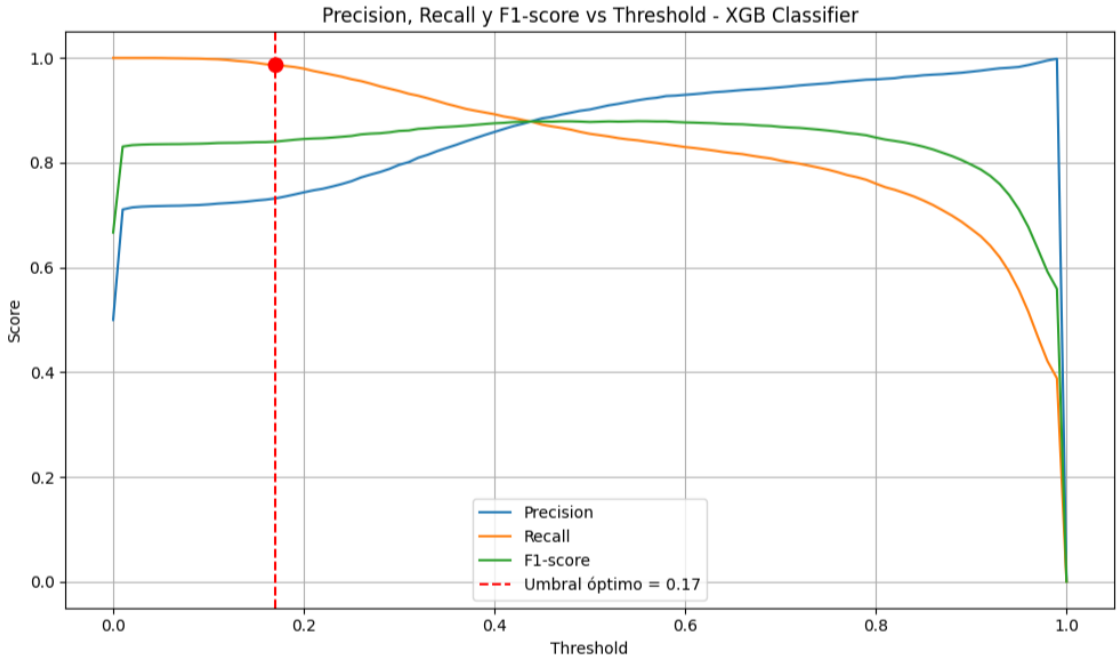

Ilustración 1 – XGBoost: Precision, Recall & F1-Score vs Threshold

Con esta gráfica seleccionamos el threshold óptimo para el modelo. La elección se basa, presumiblemente, en encontrar el punto donde el F1-score es alto y el recall es muy elevado, que es crítico en detección de fraude, donde lo que más importa es no dejar pasar fraudes reales (es decir, minimizar falsos negativos).

Recall cercano a 1.0 en el umbral 0.17:

- Prácticamente todos los fraudes están siendo detectados.

Precision menor, pero razonable (~0.7-0.8):

- Algunos no fraudes están siendo clasificados como fraude (falsos positivos), pero eso es tolerable si se quiere evitar falsos negativos.

F1-score bien balanceado (~0.85 en ese punto):

- Confirma que hay un equilibrio sólido entre recall y precisión, lo cual es útil cuando no se quiere que la precisión sea demasiado baja.

Gráfica ROC Curve:

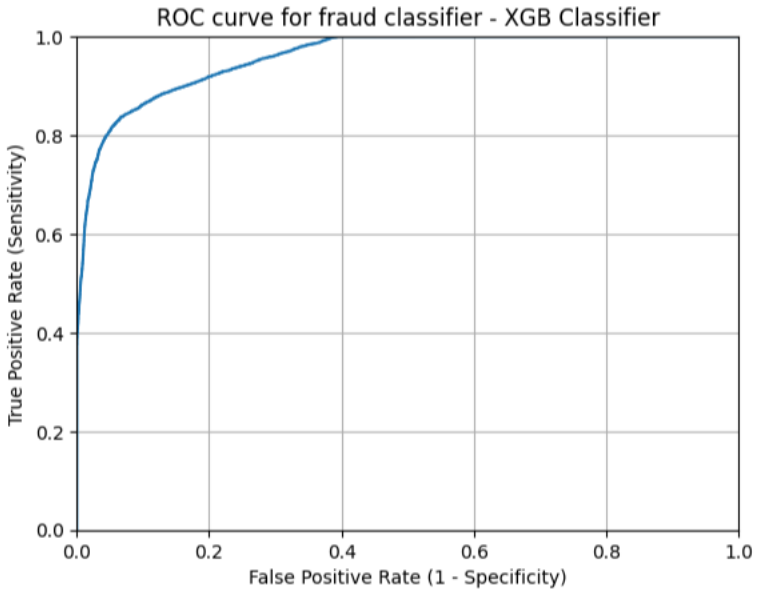

Ilustración 2 - XGBoost: ROC Curve

Curva pronunciada hacia la esquina superior izquierda:

- Significa que el modelo tiene un buen poder de discriminación entre clases.
- Puedes obtener una alta sensibilidad manteniendo una baja tasa de falsos positivos.

Área bajo la curva (ROC AUC):

- AUC ~0.95 lo que indica un modelo excelente.

Gráfico de datos reales vs. predichos

La siguiente visualización muestra una muestra de las 100 primeras predicciones del modelo XGBoost frente a las etiquetas reales:

- Azul (círculos): valores reales
- Amarillo (triángulos): valores predichos
- Rojo (cruces): error al detectar el fraude

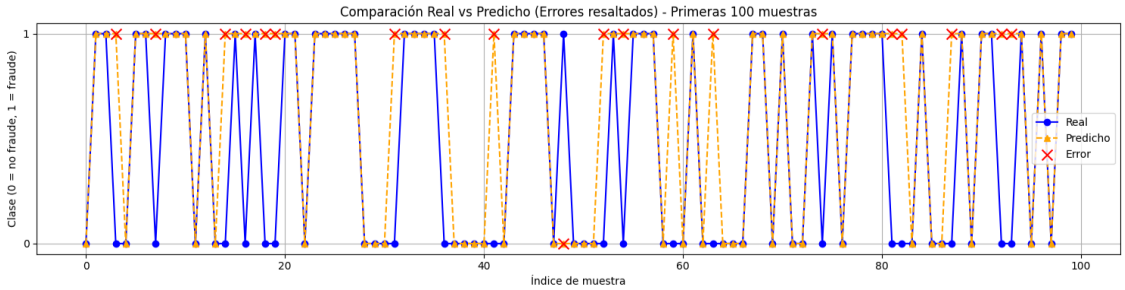

Ilustración 3 - XGBoost: Datos Reales vs Predichos

Esto demuestra que el modelo logra capturar patrones de fraude con alta fidelidad, aunque con ligeros márgenes de error aceptables en contextos de detección temprana.

Ejemplos de predicciones

Primeras 25 muestras:

True: [0 1 1 0 0 1 1 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1]

Pred: [0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1]

Como vemos el modelo no tiene ningún error al detectar el fraude, pero comete un 24% de Falsos Positivos, es decir, detecta fraude cuando no lo es.

En situaciones de fraude esta tasa controlada de falsos positivos es aceptable, ya que son entornos críticos donde el objetivo es maximizar la detección y luego refinar con procesos humanos.

4.6.2 Evaluación comparativa de modelos no supervisados para la clasificación de fraude

Para la clasificación de fraude usaremos Clustering para separar los datos en diferentes grupos y definir el tipo de fraude.

Seguidamente usaremos Autoencoders para hacer más robusto el modelo e identificar anomalías.

Por último, usaremos Isolation Forest dentro de cada clúster para detectar outliers, comportamientos anómalos dentro del propio perfil de ese tipo de fraude.

Clustering

Clustering nos ayuda a diferenciar el fraude en diferentes grupos.

En este caso se detecta 4 clústeres:

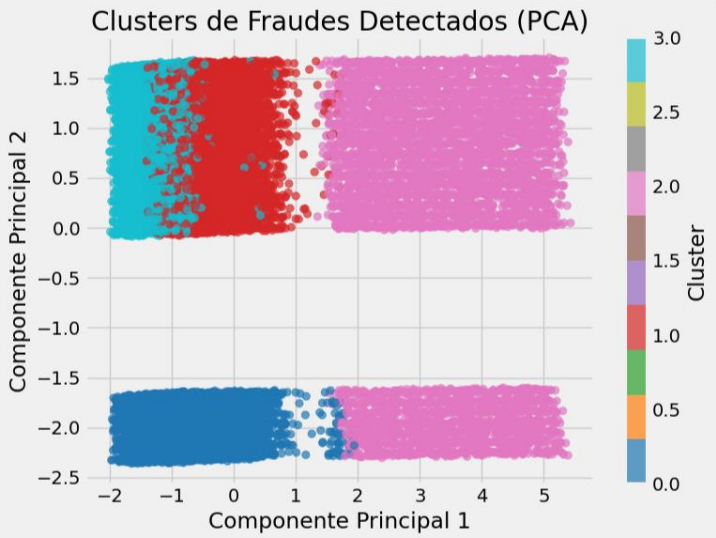

Ilustración 4 - Clustering: Grupos Detectados

clusters	duracion_seg	costo_usd	hora	es_madrugada	llamada_ internacional	destino_ premium
0	89.398390	2.838	2.961	1.000	0.898045	0.012455
1	94.608699	4.727	15.043	0.000	0.896786	0.005395
2	706.445714	35.322	11.471	0.289	0.893613	1.000000
3	81.933742	0.821	14.900	0.000	0.902954	0.003923

Tabla 24 - Clusters Detectados

Con estos datos podemos, según las reglas definidas anteriormente, definir el tipo de fraude:

Cluster	Duración (s)	Costo (USD)	Hora media	Madrugada	Llamada Intl.	Premium	Posible tipo de fraude
0	~89	~2.84	~03:00 (madrugada)	✓	✓	✗	Wangiri
1	~95	~4.72	~15:00	✗	✓	✗	General llamadas internacionales de día
2	~706	~35.3	~11:30	✗	✓	✓	IRSF (premium)

Cluster	Duración (s)	Costo (USD)	Hora media	Madrugada	Llamada Intl.	Premium	Posible tipo de fraude
3	~82	~0.82	~15:00	✗	✓	✗	SIM Box / Bypass

Tabla 25 - Clusters Detectados - Posible Fraude

- Clúster 0: Wangiri
  Llamadas muy cortas (~1.5 min), en madrugada, con origen internacional.

  Esto apunta a llamadas falsas para provocar devolución (Wangiri).
- Clúster 1: General Intl. Fraude
  Duración y costo moderados, de día, no premium.

  Podría ser uso irregular pero no grave. Tal vez CLI Spoofing si el número origen no cuadra.
- Clúster 2: IRSF
  Llamadas largas (~12 min), alto costo, destinos premium.

  Claro patrón de fraude de ingresos (International Revenue Share Fraud).
- Clúster 3: SIM Box / Bypass
  Llamadas breves, baratas, internacionales, en horario laboral. Podrían enmascarar tráfico internacional como local.

Una vez que hemos detectado que tipo de fraude es cada clúster, los etiquetamos y así podemos ver que cantidad de fraude tenemos para cada tipo:

Clúster	Tipo de fraude	Conteo
3	SIM Box / Bypass	8666
1	Fraude General Internacional	8526
0	Wangiri	6905
2	IRSF (premium)	5903

Tabla 26 - Clusters: Conteo de Fraude

Autoencoder

Aplicamos Autoencoder para hacer el modelo más robusto.

Un Autoencoder se compone de dos partes:

1. Encoder: Reduce la dimensionalidad.
2. Decoder: Reconstruye la entrada original.

A partir del error de reconstrucción se crea una etiqueta binaria llamada “fraude_autoencoder”:

- Fraude_autoencoder = 0 --> Si el error de reconstrucción es bajo, el patrón es normal. El fraude se puede clasificar según los tipos definidos.
- Fraude_autoencoder = 1 --> Si el error es alto, significa que los datos no se parecen a los patrones que el autoencoder aprendió. No se puede clasificar según los tipos definidos, por lo tanto, el autoencoder ha detectado un comportamiento fuera de lo normal (anomalía).

Detectado por Autoencoder	0	1
Cluster		
0	6585	320
1	7975	551
2	5603	300
3	8337	329

Tabla 27 - Autoencoders: Clusters Detectados

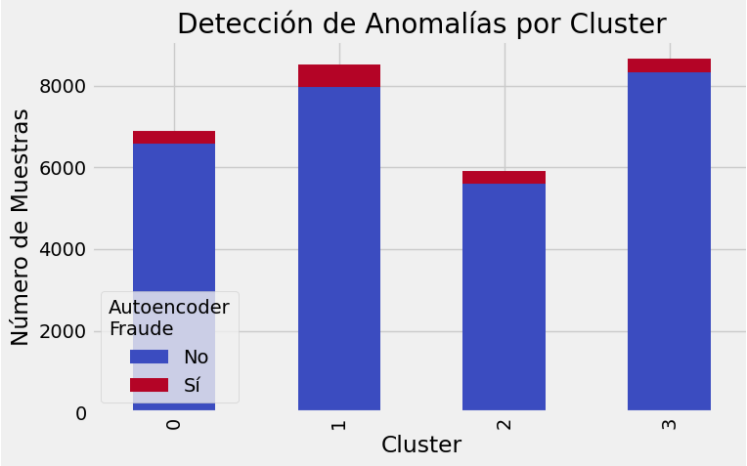

Ilustración 5 - Autoencoders: Clusters Detectados

Ejemplos de datos:

Wangiri

	usuario_id	*duracion_seg*	*numero_destino*	*pais_origen*	*pais_destino*	*tipo_llamada*	*costo_usd*	*hora*	*dia_semana*
3	8133274269269	93.23	10153	0	2	1	0.93	2	3
16	6819198836143	5.00	2762	8	3	0	0.25	6	3

Tabla 28 - Ejemplo Datos: Wangiri

IRSF (*premium*)

	usuario_id	*duracion_seg*	*numero_destino*	*pais_origen*	*pais_destino*	*tipo_llamada*	*costo_usd*	*hora*	*dia_semana*
4	8719553642652	333.11	85796	4	7	0	16.66	2	3
17	8382484970829	675.62	87020	8	9	0	33.78	18	3

Tabla 29 - Ejemplo Datos: IRSF (Premium)

Fraude General Internacional

	usuario_id	*duracion_seg*	*numero_destino*	*pais_origen*	*pais_destino*	*tipo_llamada*	*costo_usd*	*hora*	*dia_semana*
6	2468952972968	5.0	52234	7	6	0	0.25	23	1
21	1859357747201	5.0	41870	4	4	0	0.25	10	0

Tabla 30 - Ejemplo Datos: Fraude General Internacional

SIM Box / Bypass

	usuario_id	*duracion_seg*	*numero_destino*	*pais_origen*	*pais_destino*	*tipo_llamada*	*costo_usd*	*hora*	*dia_semana*
59	7513260374353	74.64	20664	4	1	1	0.75	21	6

6 1	90858865 83576	49.30	62759	9	8	1	0.49	17	6

Tabla 31 - Ejemplo Datos: SIM Box/Bypass

Isolation Forest

El Isolation Forest nos ayuda a refinar aún más la detección y puede identificar nuevas variantes de un fraude conocido.

Con esta información podemos:

1. Mostrar una comparación de diferencias más significativas:

   En este caso comparando los outliers y los valores normales podemos mostrar el top de diferencias más significativas:

	outliers	normales	diferencia_pct
llamada_internacional	0.131158	0.938662	-86.03
destino_premium	0.285619	0.197909	44.32
costo_usd	11.441678	9.065421	26.21
duracion_seg	256.209368	207.708963	23.35
s_madrugada	0.340879	0.284301	19.90

Tabla 32 - Isolation Forest: Outliers vs Valores Normales

   Valores positivos altos en diferencia_pct significan:

   - Los outliers tienen MÁS de esta característica
   - Destino_premium - los outliers van 44% más a destinos premium.
   - Costo_usd – los outlieres cuestan solo 26% más que lo normal.

   Valores negativos en diferencia_pct significan:

   - Los outliers tienen MENOS de esta característica
   - Llamada_internacional tiene -86.03%, los outliers son menos internacionales que lo normal.
   - Si normales tienen 50% internacionales, outliers solo7%.
   - El fraude es mayormente local.

   Esto sugiere:

   Posible fraude de "número premium doméstico":

- Llamadas a números 900, concursos, líneas eróticas LOCALES
- No necesitan salir del país
- Costos moderados pero repetitivos

O fraude de "desvío interno":

- Llamadas desviadas dentro del país a números controlados
- Evitan costos internacionales (menos sospechoso)
- Volúmenes medios para pasar desapercibidos

2. Detección de nuevos patrones de fraude:
   Si analizamos para encontrar nuevos patrones de fraude, el modelo encuentra 9 casos:

   Casos encontrados: 9
   Resumen de estos casos:

	clúster	duracion_seg	costo_usd	hora
Count	9.0	9.000	9.000	9.000
Mean	2.0	1192.623	59.630	1.222
Std	0.0	4.886	59.630	2.438
Min	2.0	1184.990	0.2447	0.0
25%	2.0	1190.490	59.520	0.0
50%	2.0	1191.630	59.580	0.0
75%	2.0	1196.440	59.820	0.0
max	2.0	1199.750	59.990	6.000

Tabla 33 - Nuevos Patrones de Fraude: Resumen Casos Encontrados

- Todos son del mismo clúster (2.0) lo que sugiere un perfil de comportamiento idéntico.
- La duración es uniforme, promedio de 1,192segs y desviación estándar e solo 4.9 segs. (las llamadas duran prácticamente lo mismo)
- Los costos son casi idénticos, rangos entre 59,25$ - 59.99$

- Horario muy específico, principalmente hora 0 (madrugada)

Al ser datos sintéticos, elaborados aleatoriamente, podemos ver qué tipo de patrón se ha usado.

Es interesante ver este tipo de información en datos reales para ver que nuevos fraudes podemos encontrar y crear reglas para poder evitar estas situaciones.

3. Estadísticas de anomalías y scores:

	outliders	normales
Count	1502.000	28498.00
Mean	-0.0292	0.1355
Std	0.0254	0.0528
Min	-0.1548	0.000
25%	-0.0406	0.1036
50%	-0.0226	0.1460
75%	-0.0102	0.1761
max	-0.000	0.2150

Tabla 34 - Estadísticas de Anomalías

- Separación clara de distribuciones:
  - 0% de solapamiento entre medianas.
  - Distribución de confianza:
    - 25% de outliers más extremos: score < -0.0406
    - 75% de normales más confiables: score > +0.1036

4. Distribución del score por clúster:

clusters	mean	std	min	max	count
0	0.1383	0.0719	-0.1548	0.2150	6905
1	0.1307	0.0614	-0.0974	0.2093	8526
2	0.1146	0.0556	-0.0589	0.1942	5903

3	0.1238	0.0599	-0.0969	0.1972	86666

Tabla 35 - Distribución Anomalías por Cluster

- El clúster 2 es el más peligroso:
  - Tiene la media más baja, por lo que es más anómalo
  - Tiene menos dispersión (std = 0.0556), cosa que significa que es más homogéneo
- El clúster 0 tiene más casos extremos:
  - Std = 0.0719 que indica que hay más dispersión
  - Min = -0.1548, el caso más anómalo del dataset
- Clúster 1 & 3 son intermedios con un comportamiento similar entre ellos.

5. Top de 10 casos críticos para revisión manual:
   Podemos imprimir los casos más críticos que necesitan ser validados:

	clúster	Anomaly_ score	costo_ usd	Duración_seg	hora	Es_madrugada	Llamada_ internacional	Destino_ premium
27827	0	-0.1548	14.31	286.29	0	1	0	1
18941	0	-0.1485	13.30	266.03	1	1	0	1
50443	0	-0.1414	12.14	242.72	1	1	0	1
30524	0	-0.1398	15.97	319.37	0	1	1	1
8231	0	-0.1255	8.48	169.63	0	1	0	1
56448	0	-0.1208	14.65	292.95	2	1	1	1
7517	0	-0.1194	10.7	201.45	2	1	0	1
17878	0	-0.1091	14.98	299.55	6	1	0	0
63358	0	-0.1047	13.56	271.18	1	1	1	1
33352	0	-0.1037	14.77	295.41	0	1	0	0

Tabla 36 - 10 Casos Críticos a Revisar

Con esto podemos:

1. Poner especial atención a patrones de los scores más bajos (sugieren anomalías).
2. Buscar patrones de comportamiento.

4.6.3 Conclusiones

En este apartado se presentan las principales conclusiones derivadas del desarrollo e implementación del sistema de detección de fraude en telecomunicaciones. Se sintetizan los hallazgos obtenidos a partir del análisis comparativo de modelos supervisados y no supervisados, destacando su desempeño, capacidad de detección de patrones fraudulentos y efectividad metodológica. Además, se evalúa cómo estos resultados contribuyen al entendimiento del fraude técnico, permitiendo establecer recomendaciones sobre las estrategias más eficaces para su identificación y mitigación en entornos productivos.

Desempeño de Modelos Supervisados

La investigación demostró que XGBoost fue el modelo supervisado con mejor desempeño para la detección de fraude en telecomunicaciones, superando a Random Forest, Regresión Logística y SVC. Este resultado es consistente con la literatura existente, donde los métodos de boosting han mostrado superior capacidad para:

- Manejo de características heterogéneas: Las variables de telecomunicaciones (duración, costo, ubicación, tiempo) presentan distribuciones y escalas muy diferentes.
- Captura de interacciones complejas: XGBoost identificó patrones no lineales entre variables que otros modelos no detectaron.

Efectividad del Enfoque Híbrido No Supervisado

La combinación de K-means, Autoencoders e Isolation Forest para clasificación de tipos de fraude reveló:

Detección de Patrones Complementarios:

- K-means: Identificó agrupaciones naturales por comportamiento de usuario
- Autoencoders: Capturaron anomalías en patrones temporales y de uso
- Isolation Forest: Detectó casos atípicos dentro de cada clúster con alta precisión

Descubrimiento de Tipos de Fraude:

1. Fraude Sistemático Organizado (Clúster 2):

   Llamadas internacionales de aproximadamente 20 minutos, costos uniformes alrededor de $60 y ejecución automatizada en horarios específicos, generalmente medianoche.

2. Fraude Oportunista Doméstico (Clúster 0):

   Llamadas locales a destinos premium, duración variable entre 3 y 5 minutos, concentradas en la madrugada y con costos moderados entre $8 y $16.

Validación del Enfoque Metodológico

El uso de dos etapas (supervisado → no supervisado) demostró ser altamente efectivo:

- Etapa 1 (Supervisada): XGBoost logró alta precisión en detección binaria
- Etapa 2 (No Supervisada): La combinación de técnicas permitió caracterización detallada de tipos de fraude

Los scores de anomalía mostraron separación clara entre casos normales (media: +0.1355) y fraudulentos (media: -0.0292), validando la efectividad del enfoque.

4.6.4 Discusión

En este apartado se analizan de manera crítica los hallazgos obtenidos durante el desarrollo del sistema de detección de fraude en telecomunicaciones. Se destacan las fortalezas metodológicas, los aspectos innovadores y los beneficios prácticos del enfoque híbrido supervisado/no supervisado, así como las limitaciones y desafíos inherentes al uso de datos sintéticos y modelos de machine learning. Asimismo, se abordan las implicaciones prácticas de los resultados para operadores e industria, proporcionando una base para la aplicación real y la mejora continua de estrategias de detección de fraude.

Fortalezas de la Metodología

- Enfoque Integral:

  La investigación abordó tanto la detección como la clasificación de fraude, proporcionando un sistema completo para operadores de telecomunicaciones.

- Uso de Datos Sintéticos:

  El dataset sintético permitió:

  - Control total sobre patrones de fraude implantados.
  - Evaluación objetiva sin sesgos de datos reales.
  - Reproducibilidad de experimentos.
  - Simulación de escenarios diversos de fraude.

- Complementariedad de Técnicas:
  - Modelos supervisados: Alta precisión en detección binaria.
  - Clustering: Identificación de grupos de comportamiento.
  - Autoencoders: Detección de patrones complejos no lineales.
  - Isolation Forest: Refinamiento y detección de variantes nuevas.

Limitaciones y Desafíos

- Naturaleza Sintética de los Datos:
  - Los patrones pueden no capturar toda la complejidad del fraude real.

  - Posible sobreajuste a patrones artificialmente creados.
  - Necesidad de validación con datos reales para confirmar hallazgos.
- Desbalance Inherente:
  - Riesgo de alta tasa de falsos positivos en implementación real.
  - Necesidad de ajuste continuo de thresholds.
- Evolución Temporal del Fraude:
  - Los patrones de fraude evolucionan constantemente.
  - Los modelos requieren reentrenamiento frecuente.
  - Necesidad de detección de nuevos tipos de fraude (concept drift).

Implicaciones Prácticas

- Para Operadores de Telecomunicaciones:
  1. Sistema de Alertas Multinivel: Implementar umbrales diferenciados por tipo de fraude.
  2. Monitoreo Continuo: Especial atención a horarios de madrugada y llamadas sistemáticas.
  3. Investigación Dirigida: Priorizar casos con scores < -0.04 para revisión manual.
- Para la Industria:
  - Validación de la efectividad de enfoques híbridos supervisado/no supervisado.
  - Importancia del análisis por clústeres para entender modus operandi.
  - Necesidad de sistemas adaptativos para nuevas variantes de fraude.

4.6.5 Líneas futuras de investigación

Las perspectivas de investigación futura se centran en ampliar y perfeccionar los sistemas de detección de fraude en telecomunicaciones, incorporando tanto avances metodológicos como mejoras en datos, arquitectura y evaluación. Se plantean nuevas oportunidades para integrar modelos de deep learning más sofisticados, técnicas de ensemble avanzadas y métodos de explicabilidad (XAI) que permitan interpretar decisiones complejas. Asimismo, se destacan iniciativas para enriquecer y diversificar los datos, mejorar la detección en tiempo real, y fortalecer la validación mediante métricas económicas, estabilidad temporal y simulaciones realistas. Finalmente, se considera imprescindible abordar aspectos interdisciplinarios —legales, éticos y económicos— así como fomentar la colaboración entre operadores y la creación de benchmarks abiertos, garantizando que los futuros desarrollos sean robustos, escalables y aplicables en entornos reales de telecomunicaciones.

Mejoras Metodológicas

- Modelos Avanzados de Deep Learning

    - Redes Neuronales Recurrentes (LSTM/GRU): Para capturar patrones temporales secuenciales en llamadas.
    - Transformers: Para modelar dependencias a largo plazo en comportamiento de usuarios.
    - Variational Autoencoders (VAE): Para generación de casos sintéticos más realistas.
- Ensemble Learning Avanzado
    - Stacking heterogéneo: Combinar modelos supervisados y no supervisados en meta-modelos.
    - Dynamic Ensemble: Ajuste automático de pesos según performance reciente.
    - Multi-objective Optimization: Balance automático entre precisión y recall.
- Técnicas de Explicabilidad (XAI)
    - SHAP/LIME: Para explicar decisiones de modelos complejos.
    - Rule Extraction: Convertir modelos complejos en reglas interpretables.

Datos y Características

- Enriquecimiento de Datos
    - Datos de geolocalización: Patrones espaciales de fraude.
    - Información de dispositivos: Análisis de IMEI y características técnicas.
    - Redes sociales: Grafos de relaciones entre usuarios fraudulentos.
    - Datos externos: Listas negras, indicadores económicos, eventos especiales.
- Feature Engineering Avanzado
    - Ventanas temporales dinámicas: Agregaciones adaptativas según comportamiento.
    - Características de red: Centralidad, clustering coefficient en grafos de llamadas.
    - Anomalías relativas: Desviaciones respecto al perfil histórico individual.
    - Patrones estacionales: Detección de anomalías considerando ciclos temporales.

Sistemas en Tiempo Real

- Stream Processing
    - Apache Kafka + Spark Streaming: Para procesamiento de datos en tiempo real.
    - Edge Computing: Detección distribuida en nodos de red.
    - Event-driven Architecture: Respuesta inmediata a patrones críticos.

- Aprendizaje Online
    - Incremental Learning: Actualización continua sin reentrenamiento completo.
    - Concept Drift Detection: Identificación automática de cambios en patrones.
    - Active Learning: Solicitud de etiquetas para casos inciertos.
    - Federated Learning: Aprendizaje colaborativo entre operadores.

Evaluación y Validación

- Métricas Avanzadas
    - Cost-sensitive Metrics: Considerando impacto económico real del fraude.
    - Temporal Stability: Evaluación de degradación de modelos en el tiempo.
    - Fairness Metrics: Evitar sesgos contra grupos específicos de usuarios.
    - Uncertainty Quantification: Medidas de confianza en predicciones.
- Simulación y Testing
    - Digital Twins: Simulación completa de redes de telecomunicaciones.
    - Adversarial Testing: Evaluación contra ataques adaptativos.
    - Stress Testing: Performance bajo condiciones extremas.
    - A/B Testing: Evaluación controlada de nuevos algoritmos.

Aspectos Interdisciplinarios

- Aspectos Legales y Éticos
    - Privacy-preserving ML: Técnicas que protejan datos personales.
    - Regulatory Compliance: Cumplimiento de GDPR, CCPA y regulaciones locales.
    - Algorithmic Auditing: Transparencia y rendición de cuentas.
    - Bias Detection: Identificación y mitigación de sesgos algorítmicos.
- Aspectos Económicos
    - ROI Modeling: Modelos de retorno de inversión en sistemas anti-fraude.
    - Game Theory: Modelado de interacciones estratégicas entre fraudulentos y sistemas.
    - Risk Assessment: Cuantificación probabilística de pérdidas.
    - Insurance Applications: Integración con modelos actuariales.

Colaboración y Datos Abiertos

- Iniciativas Colaborativas
    - Consorcios industriales: Compartición segura de patrones de fraude.

  - Datasets públicos: Creación de benchmarks estándar para la comunidad.
  - Competencias académicas: Challenges para impulsar innovación.
  - Estándares internacionales: Protocolos comunes de detección.
- Transferencia de Conocimiento
  - Cross-domain Learning: Aplicación de técnicas a otros tipos de fraude.
  - Domain Adaptation: Transferencia entre operadores y regiones.
  - Meta-learning: Aprendizaje rápido de nuevos tipos de fraude.
  - Knowledge Distillation: Transferencia de conocimiento entre modelos.

Capítulo 5. REFERENCIAS

Communications Fraud Control Association. (2023). *2023 Fraud loss survey*. https://www.cfca.org

ENISA. (2024). ENISA Threat Landscape 2024. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024

Subex. (2023). Telecommunications Fraud 2023: Top 10 Telecom Frauds and How to Prevent Them. https://www.subex.com/article/telecommunications-fraud-2023-top-10-telecom-frauds-and-how-to-prevent-them/

GSM Association (GSMA). (2021). Mobile Telecommunications Security Landscape. https://www.gsma.com/solutions-and-impact/technologies/security/wp-content/uploads/2021/03/GSMA-Mobile-Telcommunications-Security-Landscape-2021.pdf

Ministerio para la Transformación Digital y de la Función Pública. (2024). Consulta pública sobre medidas para combatir el fraude en las telecomunicaciones. Gobierno de España. https://portal.mineco.gob.es/

European Commission. (2023). Digital Decade: Cybersecurity and trust in telecommunications. https://digital-strategy.ec.europa.eu/

Communications Fraud Control Association (CFCA). (2023). Global fraud loss survey. https://www.cfca.org/global-fraud-loss-survey/

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. The CRISP-DM Consortium.

Becker, K., Volinsky, C., & Wilks, A. (2010). Fraud detection in telecommunications: History and lessons learned. *Technometrics, 52*(1), 20–33. https://www.researchgate.net/publication/228345401_Fraud_Detection_in_Telecommunications_History_and_Lessons_Learned

Abo Yehya, B., & Salhab, N. (2023, octubre 25). *Telecommunications Fraud Machine Learning-based Detection*. En *Proceedings of the 4th International Conference on Data Analytics for Business and Industry (ICDABI)* (pp. 656–661). IEEE. https://www.researchgate.net/publication/383177590_Telecommunications_Fraud_Machine_Learning-based_Detection

Babaei, K., Chen, Z., & Maul, T. (2020). *A Study of Fraud Types, Challenges and Detection Approaches in Telecommunication*. *Journal of Information Systems and Telecommunication, 7*(4), 248–261. https://www.researchgate.net/publication/343229226_A_Study_of_Fraud_Types_Challenges_and_Detection_Approaches_in_Telecommunication

Yang, J., Li, S., Huang, Z., & Wu, J. (2025, mayo 30). An improve fraud detection framework via dynamic representations and adaptive frequency response filter. *Scientific Reports, 15*(1), 19051.

https://www.researchgate.net/publication/392239454_An_improve_fraud_detection_framework_via_dynamic_representations_and_adaptive_frequency_response_filter/references

Pratihar, S. R., Paul, S., Dash, P. K., & Das, A. K. (2023). Fraud Analytics Using Machine-learning & Engineering on Big Data (FAME) for Telecom. *arXiv*. https://arxiv.org/abs/2311.00724

Datagate. (s.f.). *How Telecom Billing Works*. Datagate. Recuperado el 17 de junio de 2025. https://www.datagate-i.com/blog/how-telecom-billing-work/

Lifecycle Software. (s.f.). *5 types of billing in the telecom industry*. Lifecycle Software. Recuperado el 17 de junio de 2025. https://www.lifecycle-software.com/resources/5-types-of-billing-in-the-telecom-industry

SEON. (s.f.). *Fraude de telecomunicaciones y sus tipos*. SEON. Recuperado el 17 de junio de 2025. https://seon.io/es/recursos/fraude-de-telecomunicaciones-y-sus-tipos/

Subex. (s.f.). *Subex Medium Blog*. Medium. Recuperado el 17 de junio de 2025. subex.medium.com

ResearchGate. (s.f.). *Research articles on telecom fraud*. ResearchGate. Recuperado el 17 de junio de 2025, de https://www.researchgate.net.

Improving fraud detection efficiency: Leveraging machine learning strategies. (s. f.). *ResearchGate*. Recuperado de https://www.researchgate.net/publication/388517064_Improving_fraud_detection_efficiency_Leveraging_machine_learning_strategies

Boyadjiev, C., & Lyons, M. (2024, abril 30). *Building a better mousetrap for telco fraud detection begins with data monetization & strong analytics*. Protiviti. https://blog.protiviti.com/2024/04/30/building-a-better-mousetrap-for-telco-fraud-detection-begins-with-data-monetization-strong-analytics/

Communications Fraud Control Association (CFCA). (2023, noviembre 13). *Telecommunications fraud increased 12% in 2023 equating to an estimated $38.95 billion lost to fraud*. https://cfca.org/telecommunications-fraud-increased-12-in-2023-equating-to-an-estimated-38-95-billion-lost-to-fraud/

OpenAI. (2025). *ChatGPT*. https://chat.openai.com/

ANEXO 1

TFM - Deteccion de Fraude en Telecomunicaciones

September 11, 2025

1 Librerias

[ ]:
```
#Instalación de librerias
!pip install faker
!pip install imblearn
```

[ ]:
```
import pandas as pd
import numpy as np

# Creación aleatoria de datos
import random
from faker import Faker
from datetime import datetime, timedelta

# Modelado
from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier,plot_tree
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, recall_score, precision_score, roc_curve, roc_auc_score,␣
 ↪precision_recall_curve, precision_recall_fscore_support, make_scorer
import matplotlib.pyplot as plt
import seaborn as sns

import joblib
```

2 Datos

2.1 Creacion Aleatoria de Datos

[ ]:
```
# Inicialización
faker = Faker()
random.seed(42)
np.random.seed(42)

# Configuración general
n_registros = 100000
fraude_pct = 0.30
n_fraudes = int(n_registros * fraude_pct)  # 30,000 fraudes

usuarios_unicos = 500
paises = ['US', 'MX', 'BR', 'AR', 'CO', 'ES', 'FR', 'NG', 'IN', 'CN']
tipos_llamada = ['local', 'internacional', 'roaming']
prefijos_premium = ['900', '899', '808']

# Crear lista de usuarios únicos
usuarios = [faker.msisdn() for _ in range(usuarios_unicos)]

# Funciones auxiliares
def generar_numero(premium=False, spoof=False):
    if premium:
        return random.choice(prefijos_premium) + faker.msisdn()[3:12]
    if spoof:
        return faker.random_element(elements=('00123456789', '09123456789',␣
 ↪'0700112233'))
    return faker.msisdn()

def hora_random(start=None):
    if not start:
        start = faker.date_time_this_year()
    delta = timedelta(minutes=random.randint(1, 60))
    return start + delta

# Tipos de fraude simulados
fraude_tipos = ['SIM_box', 'Bypass', 'IRSF', 'Wangiri', 'CLI_Spoof']

def generar_llamada_realista(es_fraude=False, hora_base=None, usuario_id=None):
    hora = hora_random(hora_base)
    numero_origen = usuario_id or random.choice(usuarios)
    numero_destino = generar_numero()
    pais_origen = random.choice(paises)
    pais_destino = random.choice(paises)
    tipo = random.choices(tipos_llamada, weights=[0.7, 0.2, 0.1])[0]
    duracion = np.random.exponential(scale=60)
```

```
    premium = False

    if es_fraude:
        tipo_fraude = random.choice(fraude_tipos)
        if tipo_fraude == 'SIM_box':
            tipo = 'local'
            numero_origen = random.choice(usuarios)
            numero_destino = generar_numero()
            duracion = random.uniform(20, 100)
        elif tipo_fraude == 'Bypass':
            tipo = 'local'
            pais_destino = 'NG'
            duracion = random.uniform(30, 180)
        elif tipo_fraude == 'IRSF':
            tipo = 'internacional'
            numero_destino = generar_numero(premium=True)
            duracion = random.uniform(200, 1200)
        elif tipo_fraude == 'Wangiri':
            tipo = 'internacional'
            duracion = 1
        elif tipo_fraude == 'CLI_Spoof':
            tipo = 'internacional'
            numero_origen = generar_numero(spoof=True)
            duracion = random.uniform(60, 300)
    else:
        if tipo == 'internacional':
            duracion = min(duracion * 1.5, 600)

    duracion = min(max(duracion, 5), 3600)
    costo_base = 0.01 if tipo == 'local' else (0.05 if tipo == 'internacional'␣
 ↪else 0.1)
    costo = duracion * costo_base

    return {
        'usuario_id': numero_origen,
        'duracion_seg': round(duracion, 2),
        'numero_destino': numero_destino,
        'pais_origen': pais_origen,
        'pais_destino': pais_destino,
        'hora_llamada': hora.strftime("%Y-%m-%d %H:%M:%S"),
        'tipo_llamada': tipo,
        'costo_usd': round(costo, 2),
        'es_fraude': int(es_fraude)
    }

# Generación estricta del dataset
datos = []
```

```
# 1. Generar exactamente n_fraudes llamadas fraudulentas
while len(datos) < n_fraudes:
    usuario = random.choice(usuarios)
    datos.append(generar_llamada_realista(True, usuario_id=usuario))

# 2. Generar el resto como llamadas normales
while len(datos) < n_registros:
    usuario = random.choice(usuarios)
    datos.append(generar_llamada_realista(False, usuario_id=usuario))

# 3. Mezclar los datos y crear el DataFrame
random.shuffle(datos)
df = pd.DataFrame(datos)

# 4. Guardar como CSV
df.to_csv("cdr_fraude_realista_nuevo.csv", index=False)
print("Dataset generado y guardado como 'cdr_fraude_realista.csv'")
```

2.2 Descripción de los Datos

[ ]:
```
# Cargar el archivo CSV
data = pd.read_csv("cdr_fraude_realista_nuevo.csv",␣
 ↪parse_dates=['hora_llamada'])
data.head()
```

Información de los Datos

[ ]:
```
# Información general del DataFrame
print("=== Info general ===")
print(data.info())

# Estadísticas descriptivas para columnas numéricas y de tiempo
print("\n\n=== Descripción estadística ===")
print(data.describe(include='all'))
```

Conteo de Fraude

[ ]:
```
# Distribución de la variable objetivo (fraude)
print("=== Distribución de la variable 'es_fraude' ===")
print(data['es_fraude'].value_counts())
print(data['es_fraude'].value_counts(normalize=True).apply(lambda x: f"{x:.
 ↪2%}"))
```

2.2.1 Preparación de los datos

[ ]:
```
# Conversión de formato y adición de nuevas columnas
data['hora'] = data['hora_llamada'].dt.hour
data['dia_semana'] = data['hora_llamada'].dt.dayofweek
data['es_madrugada'] = data['hora'].apply(lambda x: 1 if x <= 6 else 0)
data['llamada_internacional'] = (data['pais_origen'] != data['pais_destino']).
 ↪astype(int)
data['numero_destino'] = data['numero_destino'].astype(str)
data['destino_premium'] = data['numero_destino'].str.
 ↪startswith(('900','899','808')).astype(int)
data= data.drop('hora_llamada', axis=1)
```

[ ]:
```
data.info()
```

Conversión objects a float

[ ]:
```
le= LabelEncoder()
data['numero_destino']= le.fit_transform(data['numero_destino'])
data['pais_origen']= le.fit_transform(data['pais_origen'])
data['pais_destino'] = le.fit_transform(data['pais_destino'])
data['tipo_llamada'] = le.fit_transform(data['tipo_llamada'])
data.head()
```

Balanceo de los Datos

[ ]:
```
from imblearn.under_sampling import RandomUnderSampler
rd= RandomUnderSampler()
y= data["es_fraude"]
X= data.drop("es_fraude", axis=1)
X_sampled, y_sampled= rd.fit_resample(X,y)

print("=== Distribución de la variable 'es_fraude' ===")
print(y_sampled.value_counts())
print(y_sampled.value_counts(normalize=True).apply(lambda x: f"{x:.2%}"))
```

División de los Datos en Entrenamiento y Test

[ ]:
```
X_train, X_test, y_train, y_test= train_test_split(X_sampled, y_sampled,␣
 ↪test_size= 0.3, random_state=42, stratify=y_sampled)
X_train.head()
```

Estandarización de los Datos

[ ]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3 Modelado y Evaluación

Modelos usados: - Random Forest - Logistic Regression - SVC - XGBoost

Técnicas de Evaluación: - Confussion Matrix - $Classification Accuracy$ - En general, ¿con qué frecuencia acierta el clasificador? No nos dice la distribución de los valores de las respuestas y tampoco dice que "tipos" de errores el clasifiador esta haciendo. - $Classification Error / Misclassification Rate$ - En general, ¿con qué frecuencia el clasificador se equivoca? - $Sensitivity / True Positive Rate / Recall$ - Cuando el valor actual es positivo, ¿con qué frecuencia acierta el clasificador? - $Specificity$ - Cuando el valor actual es negativo, ¿con qué frecuencia acierta el clasificador? - $False Positive Rate$ - Cuando el valor actual es negativo, ¿con qué frecuencia el clasificador se equivoca? - $Precision$ - Cuando se predice un valor positivo, ¿con qué frecuencia acierta el clasificador? La Matriz de Confusión nos permite clacular varias métricas de clasificación, y estas métricas pueden guianors a la selección del modelo. Para el presente problema "Detección de Fraude" se priorizará la optimización de la sensitividad, por que los falsos positivos (transacciones normales que son etiquetadas como posible fraude) son más acceptadas que los falsos negativos (transacciones fraudulentas que no son detectadas)

- Classification Report
- Roc_auc_score - Indica cuán bien tu modelo es capaz de distinguir entre clases (=1.0 modelo perfecto)

Se eligirá el threshold óptimo del modelo teniendo en cuenta: - $F1 - score$ - Media armónica entre precision y recall/sensitivity - $Recall / Sensitivity$

4 Random Forest

Random Forest no es un modelo sensible al escalado de los datos por lo que utilizaremos los datos sin escalar.

```
[ ]: rfc= RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred_rfc= rfc.predict(X_test)
print(accuracy_score(y_test,y_pred_rfc))
```

```
[ ]: print('Accuracy of Random Forest on training set: {:.2f}'.format(rfc.
  ↪score(X_train, y_train)))
print('Accuracy of Random Forest on test set: {:.2f}'.format(rfc.score(X_test,␣
  ↪y_test)))
```

4.1 Confussion Matrix

```
[ ]: print(confusion_matrix(y_test, y_pred_rfc))
```

4.1.1 Métricas provenientes de la Confussion Matrix

```
[ ]: print('Classification Accuracy [Logistic Regression]:', accuracy_score(y_test,␣
      ↪y_pred_rfc))
     print('Classification Error [Logistic Regression]:', (1 -␣
      ↪accuracy_score(y_test, y_pred_rfc)))
     print('Sensitivity [Logistic Regression]:', recall_score(y_test, y_pred_rfc))
     print('Specificity [Logistic Regression]:',(8166 / (8166 + 834))) # TN / (TN +␣
      ↪FP)
     print('False Positive Rate [Logistic Regression]:',(834 / (8166 + 834))) # FP /␣
      ↪(TN + FP)
     print('Precision [Logistic Regression]:', precision_score(y_test, y_pred_rfc))
```

```
[ ]: # Analizando las predicciones
     print('True:', y_test.values[0:25])
     print('Pred:', y_pred_rfc[0:25])
```

Analizando las predicciones con los valores reales nos damos cuenta de que el modelo comete varios errores al predecir el fraude. En estas 25 muestras, 3 muestras de fraude no son detectadas. Esto seria un 12%.

4.1.2 Classification Report

```
[ ]: cr_rfc= classification_report(y_test,y_pred_rfc)
     print(cr_rfc)
```

4.1.3 Ajustando el threshold del clasificador

```
[ ]: # Guardamos las probabilidades de prediccion para la clase 1
     y_pred_prob_rfc = rfc.predict_proba(X_test)[:, 1]
```

```
[ ]: # Dibujamos el histograma de las probabilidades
     fig, ax = plt.subplots()
     ax.hist(y_pred_prob_rfc, bins=8)
     ax.set_xlim(0, 1)
     ax.set_title('Histogram of predicted probabilities - RFC')
     ax.set_xlabel('Predicted probability of fraud')
     ax.set_ylabel('Frequency')
     plt.show()
```

El histograma nos muestra dos picos grande:

Cerca de 0 -> Muchas predicciones con baja probabilidad de fraude Cerca de 1 -> Muchas predicciones con alta probabilidad de fraude Esto sugiere que el modelo esta separando bien las clases en terminos de probabilidad. Cuando el threshold es bajo (cerca de 0.0):

Clasificas casi todo como fraude, por lo tanto el recall es alto (casi 1.0). Pero probablemente tendrás muchos falsos positivos (mala precisión). Cuando el threshold es alto (cerca de 1.0): Solo marcas

como fraude los casos más seguros, así que detectas menos fraudes recall bajo. Pero la precisión mejora (menos falsos positivos).

[ ]:
```python
# Definir rangos de thresholds
thresholds = np.arange(0, 1.01, 0.01)
precision_scores = []
recall_scores = []
f1_scores = []

# Calcular métricas para cada threshold
for t in thresholds:
    y_pred_class = (y_pred_prob_rfc >= t).astype(int)
    precision, recall, f1, _ = precision_recall_fscore_support(
        y_test, y_pred_class, average='binary', zero_division=0
    )
    precision_scores.append(precision)
    recall_scores.append(recall)
    f1_scores.append(f1)

# Encontrar el F1-score máximo
max_f1 = max(f1_scores)
print (f"Máximo F1-Score: {max_f1:.2f}")
```

[ ]:
```python
# Buscar los thresholds que cumplen F1 >= F1 máximo
valid_idxs = [i for i, f1 in enumerate(f1_scores) if f1 >= 0.86]

# Dentro de esos, elegir el que tenga mayor recall
best_idx = max(valid_idxs, key=lambda i: recall_scores[i])
best_threshold = thresholds[best_idx]

recall_score_rfc=recall_scores[best_idx]
f1_score_rfc=f1_scores[best_idx]
precision_score_rfc=precision_scores[best_idx]

# Resultado
print(f"Umbral óptimo [rfc]]: {best_threshold:.2f}")
print(f"Recall en ese umbral [rfc]: {recall_score_rfc:.3f}")
print(f"F1 en ese umbral [rfc]: {f1_score_rfc:.3f}")
print(f"Precision en ese umbral [rfc]: {precision_score_rfc:.3f}")

# Graficar
plt.figure(figsize=(10, 6))
plt.plot(thresholds, precision_scores, label='Precision')
plt.plot(thresholds, recall_scores, label='Recall')
plt.plot(thresholds, f1_scores, label='F1-score')

# Marcar el threshold óptimo
```

```
plt.axvline(best_threshold, color='red', linestyle='--', label=f'Umbral óptimo␣
 ↪= {best_threshold:.2f}')
plt.scatter(best_threshold, recall_scores[best_idx], color='red', s=80,␣
 ↪zorder=5)

plt.title('Precision, Recall y F1-score vs Threshold - Random Forest')
plt.xlabel('Threshold')
plt.ylabel('Score')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

4.1.4 ROC Curves and Area Under the Curve (AUC)

[ ]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_rfc)

fig, ax = plt.subplots()
ax.plot(fpr, tpr)
ax.set_xlim([0.0, 1.0])
ax.set_ylim([0.0, 1.0])
ax.set_title('ROC curve for fraud classifier - Random Forest')
ax.set_xlabel('False Positive Rate (1 - Specificity)')
ax.set_ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
plt.show()
```

[ ]:
```
# Función que acepta un threshold e imprime la sensitividad y la especifidad
def evaluate_threshold(threshold):
    print('Sensitivity:', tpr[thresholds > threshold][-1])
    print('Specificity:', 1 - fpr[thresholds > threshold][-1])
```

[ ]:
```
evaluate_threshold(best_threshold)
```

[ ]:
```
print('Roc auc score [Random Forest]:',roc_auc_score(y_test, y_pred_prob_rfc))
```

4.2 Modelo ajustado al threshold más óptimo

[ ]:
```
# Obtenemos métrics con el modelo ajustado al mejor threshold
model = RandomForestClassifier()
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

accuracy_scores = []
recall_scores = []
y_train = y_train.values if hasattr(y_train, "values") else y_train  #␣
 ↪convertir pandas Series a ndarray
```

```
for train_idx, val_idx in kf.split(X_train, y_train):
    X_tr, X_val = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_tr, y_val = y_train[train_idx], y_train[val_idx]

    model.fit(X_tr, y_tr)
    probs = model.predict_proba(X_val)[:, 1]
    preds = (probs >= best_threshold).astype(int)

    accuracy_scores.append(accuracy_score(y_val, preds))
    recall_scores.append(recall_score(y_val, preds))

recall_score_rfc_1 = np.mean(recall_scores)

print(f"Accuracy promedio: {np.mean(accuracy_scores):.4f} ± {np.
 ↪std(accuracy_scores):.4f}")
print(f"Recall promedio: {recall_score_rfc_1:.4f} ± {np.std(recall_scores):.
 ↪4f}")
```

[ ]:
```
# Predicciones con el threshold optimo (0.33)
y_pred_rfc_pred = (y_pred_prob_rfc >= 0.33).astype(int)

# Convertimos y_test en un array
y_test_real = np.array(y_test)
```

[ ]:
```
# Seleccionar primeras 100 muestras
y_test_real_100 = y_test_real[:100]
y_pred_rfc_pred_100 = y_pred_rfc_pred[:100]
x = np.arange(100)

# Calcular índices donde hay error (mismatches)
errores = y_test_real_100 != y_pred_rfc_pred_100

plt.figure(figsize=(15, 4))

# Valores reales
plt.plot(x, y_test_real_100, label='Real', marker='o', linestyle='-',␣
 ↪color='blue')

# Predicciones
plt.plot(x, y_pred_rfc_pred_100, label='Predicho', marker='^', linestyle='--',␣
 ↪color='orange')

# Resaltar errores
plt.scatter(x[errores], y_pred_rfc_pred_100[errores], color='red',␣
 ↪label='Error', zorder=5, marker='x', s=100)
```

```
plt.title('Comparación Real vs Predicho (Errores resaltados) - Primeras 100␣
 ↪muestras')
plt.xlabel('Índice de muestra')
plt.ylabel('Clase (0 = no fraude, 1 = fraude)')
plt.yticks([0, 1])
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

4.3 Logistic Regression

[ ]:
```
logreg = LogisticRegression(solver='liblinear')
logreg.fit(X_train_scaled, y_train)
# Predicciones para el set de testing
y_pred_logreg = logreg.predict(X_test_scaled)
```

[ ]:
```
print('Precisión [Logistic Regression] en el conjunto de entrenamiento: {:.2f}'.
 ↪format(logreg.score(X_train_scaled, y_train)))
print('Precisión [Logistic Regression] en el conjunto de test: {:.2f}'.
 ↪format(logreg.score(X_test_scaled, y_test)))
```

4.3.1 Confussion Matrix

[ ]:
```
print(confusion_matrix(y_test, y_pred_logreg))
```

Metricas provenientes de la Confussion Matrix

[ ]:
```
print('Classification Accuracy [Logistic Regression]:', accuracy_score(y_test,␣
 ↪y_pred_logreg))
print('Classification Error [Logistic Regression]:', (1 -␣
 ↪accuracy_score(y_test, y_pred_logreg)))
print('Sensitivity [Logistic Regression]:', recall_score(y_test, y_pred_logreg))
print('Specificity [Logistic Regression]:',(6696 / (6696 + 2304))) # TN / (TN +␣
 ↪FP)
print('False Positive Rate [Logistic Regression]:',(2304 / (6696 + 2304))) # FP␣
 ↪/ (TN + FP)
print('Precision [Logistic Regression]:', precision_score(y_test,␣
 ↪y_pred_logreg))
```

[ ]:
```
# Analizando las predicciones
print('True:', y_test.values[0:25])
print('Pred:', y_pred_logreg[0:25])
```

Analizando las predicciones con los valores reales nos damos cuenta de que el modelo comete varios errores al predecir el fraude. En estas 25 muestras, 5 muestras de fraude no son detectadas. Esto seria un 20%.

4.3.2 Classification Report

```
[ ]: cr_logreg= classification_report(y_test,y_pred_logreg)
print(cr_logreg)
```

4.3.3 Ajustando el threshold del clasificador

```
[ ]: # Guardamos las probabilidades de prediccion para la clase 1
y_pred_prob = logreg.predict_proba(X_test_scaled)[:, 1]
```

```
[ ]: # Dibujamos el histograma de las probabilidades
fig, ax = plt.subplots()
ax.hist(y_pred_prob, bins=8)
ax.set_xlim(0, 1)
ax.set_title('Histogram of predicted probabilities - Logistic Regression')
ax.set_xlabel('Predicted probability of fraud')
ax.set_ylabel('Frequency')
plt.show()
```

- Hay una distribucion relativamente uniforme: Las probabilidades están bastante distribuidas entre 0 y 1, sin una fuerte concentración en un solo extremo. Eso sugiere que el modelo está indeciso en muchos casos. Es decir, no está prediciendo con mucha certeza qué es fraude y qué no.
- Mucha probabilidad en el centro (0.3 – 0.7): Hay un gran número de predicciones en el rango medio. Esto es una señal de que el modelo no está completamente seguro y puede que le cueste separar bien las clases.
- No hay picos fuertes en los extremos (0 o 1): Un buen modelo para clasificación binaria muchas veces muestra acumulación en los extremos (0 o 1), cuando está seguro. Aquí no es el caso, lo que puede indicar que hay ambigüedad en los datos o que el modelo podría beneficiarse de ingeniería de características o ajuste de parámetros.

```
[ ]: # Definir rangos de thresholds
thresholds = np.arange(0, 1.01, 0.01)
precision_scores = []
recall_scores = []
f1_scores = []

# Calcular métricas para cada threshold
for t in thresholds:
    y_pred_class = (y_pred_prob >= t).astype(int)
    precision, recall, f1, _ = precision_recall_fscore_support(
        y_test, y_pred_class, average='binary', zero_division=0
    )
    precision_scores.append(precision)
    recall_scores.append(recall)
    f1_scores.append(f1)
```

```
# Encontrar el F1-score máximo
max_f1 = max(f1_scores)
print (f"Máximo F1-Score: {max_f1:.2f}")
```

[ ]:
```
# Buscar los thresholds que cumplen F1 >= F1 máximo
valid_idxs = [i for i, f1 in enumerate(f1_scores) if f1 >= 0.74]

# Dentro de esos, elegir el que tenga mayor recall
best_idx = max(valid_idxs, key=lambda i: recall_scores[i])
best_threshold = thresholds[best_idx]

recall_score_logreg=recall_scores[best_idx]
f1_score_logreg=f1_scores[best_idx]
precision_score_logreg=precision_scores[best_idx]

# Resultado
print(f"Umbral óptimo: {best_threshold:.2f}")
print(f"Recall en ese umbral: {recall_score_logreg:.3f}")
print(f"F1 en ese umbral: {f1_score_logreg:.3f}")
print(f"Precision en ese umbral: {precision_score_logreg:.3f}")

# Graficar
plt.figure(figsize=(10, 6))
plt.plot(thresholds, precision_scores, label='Precision')
plt.plot(thresholds, recall_scores, label='Recall')
plt.plot(thresholds, f1_scores, label='F1-score')

# Marcar el threshold óptimo
plt.axvline(best_threshold, color='red', linestyle='--', label=f'Umbral óptimo␣
 ↪= {best_threshold:.2f}')
plt.scatter(best_threshold, recall_scores[best_idx], color='red', s=80,␣
 ↪zorder=5)

plt.title('Precision, Recall y F1-score vs Threshold - Logistic Regression')
plt.xlabel('Threshold')
plt.ylabel('Score')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

4.3.4 ROC Curves and Area Under the Curve (AUC)

[ ]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

fig, ax = plt.subplots()
ax.plot(fpr, tpr)
```

```
ax.set_xlim([0.0, 1.0])
ax.set_ylim([0.0, 1.0])
ax.set_title('ROC curve for fraud classifier - Logistic Regression')
ax.set_xlabel('False Positive Rate (1 - Specificity)')
ax.set_ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
plt.show()
```

[ ]:
```
# Función que acepta un threshold e imprime la sensitividad y la especifidad
def evaluate_threshold(threshold):
    print('Sensitivity:', tpr[thresholds > threshold][-1])
    print('Specificity:', 1 - fpr[thresholds > threshold][-1])
```

[ ]:
```
evaluate_threshold(best_threshold)
```

[ ]:
```
print('Roc auc score [Logistic Regression]:',roc_auc_score(y_test, y_pred_prob))
```

4.4 Modelo ajustado al threshold más óptimo

[ ]:
```
# Obtenemos métrics con el modelo ajustado al mejor threshold
model = LogisticRegression()
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

accuracy_scores = []
recall_scores = []
y_train = y_train.values if hasattr(y_train, "values") else y_train  #␣
 ↪convertir pandas Series a ndarray

for train_idx, val_idx in kf.split(X_train_scaled, y_train):
    X_tr, X_val = X_train_scaled[train_idx], X_train_scaled[val_idx]
    y_tr, y_val = y_train[train_idx], y_train[val_idx]

    model.fit(X_tr, y_tr)
    probs = model.predict_proba(X_val)[:, 1]
    preds = (probs >= best_threshold).astype(int)

    accuracy_scores.append(accuracy_score(y_val, preds))
    recall_scores.append(recall_score(y_val, preds))

recall_score_logreg = np.mean(recall_scores)

print(f"Accuracy promedio: {np.mean(accuracy_scores):.4f} ± {np.
 ↪std(accuracy_scores):.4f}")
print(f"Recall promedio: {recall_score_logreg:.4f} ± {np.std(recall_scores):.
 ↪4f}")
```

4.5 SVC (Support Vector Classifier)

[ ]:
```
# Buscamos los mejores parametros con GridSearch
grid = GridSearchCV(estimator=SVC(),
             param_grid={'C': [1, 10, 15], 'kernel': ['linear', 'rbf']})
```

[ ]:
```
grid.fit(X_train_scaled, y_train)
```

[ ]:
```
# Mejores parametros
grid.best_estimator_
```

[ ]:
```
pd.DataFrame(grid.cv_results_)[['mean_test_score', 'std_test_score', 'params']]
```

[ ]:
```
# Mejor score
grid.best_score_
```

[ ]:
```
#Dibujamos los resultados
results = pd.DataFrame(grid.cv_results_)
sns.lineplot(x='param_C', y='mean_test_score', hue='param_kernel', data=results)
plt.title("Performance de SVC según C y kernel")
plt.xscale('log')  # útil si C varía exponencialmente
plt.show()
```

Como se puede observar en los resultados los mejores parametros son: - Kernel: 'rbf' - C: '15' La diferencia entre C=10 ay C=15 no es muy grande, por lo tanto cogeremos C=10 ya que tiene menos coste computacional.

[ ]:
```
# Entrenamos el modelo con los parametros elegidos
svc= SVC(kernel='rbf', C=10, probability=True)
svc.fit(X_train_scaled, y_train)
y_pred_svc = svc.predict(X_test_scaled)
print(accuracy_score(y_test, y_pred_svc))
```

[ ]:
```
print('Accuracy of SVC on training set: {:.2f}'.format(svc.
 ↪score(X_train_scaled, y_train)))
print('Accuracy of SVC on test set: {:.2f}'.format(svc.score(X_test_scaled,␣
 ↪y_test)))
```

4.5.1 Confussion Matrix

[ ]:
```
print(confusion_matrix(y_test, y_pred_svc))
```

4.5.2 Metricas provenientes de la Confussion Matrix

[ ]:
```
print('Classification Accuracy [SVC]:', accuracy_score(y_test, y_pred_svc))
print('Classification Error [SVC]:', (1 - accuracy_score(y_test, y_pred_svc)))
print('Sensitivity [SVC]:', recall_score(y_test, y_pred_svc))
print('Specificity [SVC]:',(7796 / (7796 + 1204))) # TN / (TN + FP)
```

```
print('False Positive Rate [SVC]:',(1204 / (7796 + 1204))) # FP / (TN + FP)
print('Precision [SVC]:', precision_score(y_test, y_pred_svc))
```

[ ]:
```
# Analizando las predicciones
print('True:', y_test.values[0:25])
print('Pred:', y_pred_svc[0:25])
```

Analizando las predicciones con los valores reales nos damos cuenta de que el modelo comete varios errores al predecir el fraude. En estas 25 muestras, 3 muestras de fraude no son detectadas. Esto seria un 12%.

4.5.3 Classification Report

[ ]:
```
cr_svc= classification_report(y_test,y_pred_svc)
print(cr_svc)
```

4.6 Ajustando el threshold del clasificador

[ ]:
```
y_pred_prob_svc = svc.predict_proba(X_test_scaled)[:, 1]
```

[ ]:
```
# histogram of predicted probabilities
fig, ax = plt.subplots()
ax.hist(y_pred_prob_svc, bins=8)
ax.set_xlim(0, 1)
ax.set_title('Histogram of predicted probabilities - SVC')
ax.set_xlabel('Predicted probability of fraud')
ax.set_ylabel('Frequency')
plt.show()
```

El histograma nos muestra dos picos grande: - Cerca de 0 -> Muchas predicciones con baja probabilidad de fraude - Cerca de 1 -> Muchas predicciones con alta probabilidad de fraude Esto sugiere que el modelo esta separando bien las clases en terminos de probabilidad.

Cuando el threshold es bajo (cerca de 0.0): - Clasificas casi todo como fraude, por lo tanto el recall es alto (casi 1.0). - Pero probablemente tendrás muchos falsos positivos (mala precisión). Cuando el threshold es alto (cerca de 1.0): - Solo marcas como fraude los casos más seguros, así que detectas menos fraudes recall bajo. - Pero la precisión mejora (menos falsos positivos).

Hay que evaluar el trade-off: - Maximizar la sensitividad (recall) para no dejar fraudes sin detectar. - Especificidad baja (mas falsos positivos) - Esto seria un buen intercambio en contextos de fraude, donde es mejor investigar un falso positivo que perder un fraude real, pero con un threshold demasiado bajo te inundas de falsos positivos, lo que puede ser costoso. - Hay que elegir un threshold óptimo que te dé alto recall sin sacrificar demasiado la precisión.

[ ]:
```
# Definir rangos de thresholds
thresholds = np.arange(0, 1.01, 0.01)
precision_scores = []
recall_scores = []
f1_scores = []
```

```
# Calcular métricas para cada threshold
for t in thresholds:
    y_pred_class = (y_pred_prob_svc >= t).astype(int)
    precision, recall, f1, _ = precision_recall_fscore_support(
        y_test, y_pred_class, average='binary', zero_division=0
    )
    precision_scores.append(precision)
    recall_scores.append(recall)
    f1_scores.append(f1)

# Encontrar el F1-score máximo
max_f1 = max(f1_scores)
print (f"Máximo F1-Score: {max_f1:.2f}")
```

[ ]:
```
# Buscar los thresholds que cumplen F1 >= F1 máximo
valid_idxs = [i for i, f1 in enumerate(f1_scores) if f1 >= 0.82]

# Dentro de esos, elegir el que tenga mayor recall
best_idx = max(valid_idxs, key=lambda i: recall_scores[i])
best_threshold = thresholds[best_idx]

recall_score_svc=recall_scores[best_idx]
f1_score_svc=f1_scores[best_idx]
precision_score_svc=precision_scores[best_idx]

# Resultado
print(f"Umbral óptimo: {best_threshold:.2f}")
print(f"Recall en ese umbral: {recall_score_svc:.3f}")
print(f"F1 en ese umbral: {f1_score_svc:.3f}")
print(f"Precision en ese umbral: {precision_score_svc:.3f}")

# Graficar
plt.figure(figsize=(10, 6))
plt.plot(thresholds, precision_scores, label='Precision')
plt.plot(thresholds, recall_scores, label='Recall')
plt.plot(thresholds, f1_scores, label='F1-score')

# Marcar el threshold óptimo
plt.axvline(best_threshold, color='red', linestyle='--', label=f'Umbral óptimo␣
 ↪= {best_threshold:.2f}')
plt.scatter(best_threshold, recall_scores[best_idx], color='red', s=80,␣
 ↪zorder=5)

plt.title('Precision, Recall y F1-score vs Threshold - SVC')
plt.xlabel('Threshold')
plt.ylabel('Score')
```

```
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

4.6.1 ROC Curve and Area Under the Curve (AUC)

[ ]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_svc)

fig, ax = plt.subplots()
ax.plot(fpr, tpr)
ax.set_xlim([0.0, 1.0])
ax.set_ylim([0.0, 1.0])
ax.set_title('ROC curve for fraud classifier - SVC')
ax.set_xlabel('False Positive Rate (1 - Specificity)')
ax.set_ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
plt.show()
```

[ ]:
```
# Función que acepta un threshold e imprime la sensitividad y la especifidad
def evaluate_threshold(threshold):
    print('Sensitivity:', tpr[thresholds > threshold][-1])
    print('Specificity:', 1 - fpr[thresholds > threshold][-1])
```

[ ]:
```
evaluate_threshold(best_threshold)
```

[ ]:
```
print('Roc auc score [SVC]:',roc_auc_score(y_test, y_pred_prob_svc))
```

4.7 Modelo ajustado al threshold más óptimo

[ ]:
```
# Obtenemos métrics con el modelo ajustado al mejor threshold
model= SVC(kernel='rbf', C=10, probability=True, random_state=42)
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

accuracy_scores = []
recall_scores = []
y_train = y_train.values if hasattr(y_train, "values") else y_train  #␣
 ↪convertir pandas Series a ndarray

for train_idx, val_idx in kf.split(X_train_scaled, y_train):
    X_tr, X_val = X_train_scaled[train_idx], X_train_scaled[val_idx]
    y_tr, y_val = y_train[train_idx], y_train[val_idx]

    model.fit(X_tr, y_tr)
    probs = model.predict_proba(X_val)[:, 1]
    preds = (probs >= best_threshold).astype(int)
```

```
    accuracy_scores.append(accuracy_score(y_val, preds))
    recall_scores.append(recall_score(y_val, preds))

recall_score_svc = np.mean(recall_scores)

print(f"Accuracy promedio: {np.mean(accuracy_scores):.4f} ± {np.
 ↪std(accuracy_scores):.4f}")
print(f"Recall promedio: {recall_score_svc:.4f} ± {np.std(recall_scores):.4f}")
```

4.8 XGBoost Classifier

[ ]:
```
xgc= XGBClassifier()
xgc.fit(X_train_scaled, y_train)
y_pred_xgc= xgc.predict(X_test_scaled)
print(accuracy_score(y_test, y_pred_xgc))
```

[ ]:
```
print('Accuracy of XGBoost classifier on training set: {:.2f}'.format(xgc.
 ↪score(X_train_scaled, y_train)))
print('Accuracy of XGBoost classifier on test set: {:.2f}'.format(xgc.
 ↪score(X_test_scaled, y_test)))
```

4.8.1 Confussion Matrix

[ ]:
```
print(confusion_matrix(y_test, y_pred_xgc))
```

Métricas provenientes de la Confussion Matrix

[ ]:
```
print('Classification Accuracy [Logistic Regression]:', accuracy_score(y_test,␣
 ↪y_pred_xgc))
print('Classification Error [Logistic Regression]:', (1 -␣
 ↪accuracy_score(y_test, y_pred_xgc)))
print('Sensitivity [Logistic Regression]:', recall_score(y_test, y_pred_logreg))
print('Specificity [Logistic Regression]:',(8160 / (8160 + 840))) # TN / (TN +␣
 ↪FP)
print('False Positive Rate [Logistic Regression]:',(840 / (8160 + 840))) # FP /␣
 ↪(TN + FP)
print('Precision [Logistic Regression]:', precision_score(y_test, y_pred_xgc))
```

[ ]:
```
# Analizando las predicciones
print('True:', y_test.values[0:25])
print('Pred:', y_pred_xgc[0:25])
```

Analizando las predicciones con los valores reales nos damos cuenta de que el modelo comete varios errores al predecir el fraude. En estas 25 muestras, 3 muestras de fraude no son detectadas. Esto seria un 12%.

4.8.2 Classification Report

[ ]:
```
cr_xgc= classification_report(y_test,y_pred_xgc)
print(cr_xgc)
```

4.8.3 Ajustando del threshold del clasificador

[ ]:
```
y_pred_prob_xgc = xgc.predict_proba(X_test_scaled)[:, 1]
```

[ ]:
```
# histogram of predicted probabilities
fig, ax = plt.subplots()
ax.hist(y_pred_prob_xgc, bins=8)
ax.set_xlim(0, 1)
ax.set_title('Histogram of predicted probabilities - XGB Classifier')
ax.set_xlabel('Predicted probability of fraud')
ax.set_ylabel('Frequency')
plt.show()
```

El histograma nos muestra dos picos grande: - Cerca de 0 -> Muchas predicciones con baja probabilidad de fraude - Cerca de 1 -> Muchas predicciones con alta probabilidad de fraude Esto sugiere que el modelo esta separando bien las clases en terminos de probabilidad.

Cuando el threshold es bajo (cerca de 0.0): - Clasificas casi todo como fraude, por lo tanto el recall es alto (casi 1.0). - Pero probablemente tendrás muchos falsos positivos (mala precisión). Cuando el threshold es alto (cerca de 1.0): - Solo marcas como fraude los casos más seguros, así que detectas menos fraudes recall bajo. - Pero la precisión mejora (menos falsos positivos).

[ ]:
```
# Definir rangos de thresholds
thresholds = np.arange(0, 1.01, 0.01)
precision_scores = []
recall_scores = []
f1_scores = []

# Calcular métricas para cada threshold
for t in thresholds:
    y_pred_class = (y_pred_prob_xgc >= t).astype(int)
    precision, recall, f1, _ = precision_recall_fscore_support(
        y_test, y_pred_class, average='binary', zero_division=0
    )
    precision_scores.append(precision)
    recall_scores.append(recall)
    f1_scores.append(f1)

# Encontrar el F1-score máximo
max_f1 = max(f1_scores)
print (f"Máximo F1-Score: {max_f1:.2f}")
```

[ ]:
```
# Buscar los thresholds que cumplen F1 >= F1 máximo
valid_idxs = [i for i, f1 in enumerate(f1_scores) if f1 >= 0.84]

# Dentro de esos, elegir el que tenga mayor recall
best_idx = max(valid_idxs, key=lambda i: recall_scores[i])
best_threshold = thresholds[best_idx]

recall_score_xgc=recall_scores[best_idx]
f1_score_xgc=f1_scores[best_idx]
precision_score_xgc=precision_scores[best_idx]

# Resultado
print(f"Umbral óptimo: {best_threshold:.2f}")
print(f"Recall en ese umbral: {recall_score_xgc:.3f}")
print(f"F1 en ese umbral: {f1_score_xgc:.3f}")
print(f"Precision en ese umbral: {precision_score_xgc:.3f}")

# Graficar
plt.figure(figsize=(10, 6))
plt.plot(thresholds, precision_scores, label='Precision')
plt.plot(thresholds, recall_scores, label='Recall')
plt.plot(thresholds, f1_scores, label='F1-score')

# Marcar el threshold óptimo
plt.axvline(best_threshold, color='red', linestyle='--', label=f'Umbral óptimo␣
 ↪= {best_threshold:.2f}')
plt.scatter(best_threshold, recall_scores[best_idx], color='red', s=80,␣
 ↪zorder=5)

plt.title('Precision, Recall y F1-score vs Threshold - XGB Classifier')
plt.xlabel('Threshold')
plt.ylabel('Score')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

4.8.4 ROC Curve and Area Under the Curve (AUC)

[ ]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_xgc)

fig, ax = plt.subplots()
ax.plot(fpr, tpr)
ax.set_xlim([0.0, 1.0])
ax.set_ylim([0.0, 1.0])
ax.set_title('ROC curve for fraud classifier - XGB Classifier')
ax.set_xlabel('False Positive Rate (1 - Specificity)')
```

```
ax.set_ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
plt.show()
```

[ ]:
```
# Función que acepta un threshold e imprime la sensitividad y la especifidad
def evaluate_threshold(threshold):
    print('Sensitivity:', tpr[thresholds > threshold][-1])
    print('Specificity:', 1 - fpr[thresholds > threshold][-1])
```

[ ]:
```
evaluate_threshold(best_threshold)
```

[ ]:
```
print('Roc auc score [XGB Classifier]:',roc_auc_score(y_test, y_pred_prob_xgc))
```

4.9 Modelo ajustado al threshold más óptimo

[ ]:
```
# Obtenemos métrics con el modelo ajustado al mejor threshold
model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

accuracy_scores = []
recall_scores = []
y_train = y_train.values if hasattr(y_train, "values") else y_train  #␣
 ↪convertir pandas Series a ndarray

for train_idx, val_idx in kf.split(X_train_scaled, y_train):
    X_tr, X_val = X_train_scaled[train_idx], X_train_scaled[val_idx]
    y_tr, y_val = y_train[train_idx], y_train[val_idx]

    model.fit(X_tr, y_tr)
    probs = model.predict_proba(X_val)[:, 1]
    preds = (probs >= best_threshold).astype(int)

    accuracy_scores.append(accuracy_score(y_val, preds))
    recall_scores.append(recall_score(y_val, preds))

recall_score_xgc = np.mean(recall_scores)

print(f"Accuracy promedio: {np.mean(accuracy_scores):.4f} ± {np.
 ↪std(accuracy_scores):.4f}")
print(f"Recall promedio: {recall_score_xgc:.4f} ± {np.std(recall_scores):.4f}")
```

5 Modelo Elegido - XGB Classifier

[ ]:
```
# Predicciones con el threshold optimo (0.19)
y_pred_xgc_pred = (y_pred_prob_xgc >= 0.19).astype(int)

# Convertimos y_test en un array
y_test_real = np.array(y_test)
```

[ ]:
```
print('True:', y_test_real[:25])
print('Pred:', y_pred_xgc_pred[:25])
```

[ ]:
```
# Seleccionar primeras 100 muestras
y_test_real_100 = y_test_real[:100]
y_pred_xgc_pred_100 = y_pred_xgc_pred[:100]
x = np.arange(100)

# Calcular índices donde hay error (mismatches)
errores = y_test_real_100 != y_pred_xgc_pred_100

plt.figure(figsize=(15, 4))

# Valores reales
plt.plot(x, y_test_real_100, label='Real', marker='o', linestyle='-',␣
 ↪color='blue')

# Predicciones
plt.plot(x, y_pred_xgc_pred_100, label='Predicho', marker='^', linestyle='--',␣
 ↪color='orange')

# Resaltar errores
plt.scatter(x[errores], y_pred_xgc_pred_100[errores], color='red',␣
 ↪label='Error', zorder=5, marker='x', s=100)

plt.title('Comparación Real vs Predicho (Errores resaltados) - Primeras 100␣
 ↪muestras')
plt.xlabel('Índice de muestra')
plt.ylabel('Clase (0 = no fraude, 1 = fraude)')
plt.yticks([0, 1])
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

6 Guardar el modelo

[ ]:
```
joblib.dump(model, 'xgb_classifier_model.pkl')
```

ANEXO 2

TFM_Fraud Classification

September 11, 2025

[ ]:
```
import pandas as pd
import numpy as np

# Modelado
from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier,plot_tree
from xgboost import XGBClassifier

from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, recall_score, precision_score, roc_curve, roc_auc_score,␣
 ↪precision_recall_curve, precision_recall_fscore_support, make_scorer
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
# Cargar el archivo CSV
data = pd.read_csv("cdr_fraude_realista_nuevo.csv",␣
 ↪parse_dates=['hora_llamada'])

# Conversión de formato y adición de nuevas columnas
data['hora'] = data['hora_llamada'].dt.hour
data['dia_semana'] = data['hora_llamada'].dt.dayofweek
data['es_madrugada'] = data['hora'].apply(lambda x: 1 if x <= 6 else 0)
data['llamada_internacional'] = (data['pais_origen'] != data['pais_destino']).
 ↪astype(int)
data['numero_destino'] = data['numero_destino'].astype(str)
data['destino_premium'] = data['numero_destino'].str.
 ↪startswith(('900','899','808')).astype(int)
```

```
data= data.drop('hora_llamada', axis=1)

le= LabelEncoder()
data['numero_destino']= le.fit_transform(data['numero_destino'])
data['pais_origen']= le.fit_transform(data['pais_origen'])
data['pais_destino'] = le.fit_transform(data['pais_destino'])
data['tipo_llamada'] = le.fit_transform(data['tipo_llamada'])
```

0.1 Clasificación del fraude con Reglas

[ ]:
```
def clasificar_tipo_fraude_avanzado(row):
    if row['es_fraude'] == 0:
        return 'Normal'

    # WANGIRI: llamadas muy cortas, bajo costo
    if row['duracion_seg'] <= 2 and row['costo_usd'] < 0.05:
        return 'Wangiri'

    # IRSF: número premium, largo tiempo y alto costo
    if row['destino_premium'] and row['duracion_seg'] > 300 and␣
 ↪row['costo_usd'] > 3:
        return 'IRSF (Premium)'

    # SIM BOX / BYPASS: llamada local pero destino internacional
    if not row['llamada_internacional'] and row['pais_origen'] !=␣
 ↪row['pais_destino']:
        return 'SIM Box / Bypass'

    # CLI Spoofing: inconsistencia entre país origen y código del número
    if isinstance(row['numero_destino'], str) and not row['numero_destino'].
 ↪startswith(row['pais_origen'][:2]):
        return 'CLI Spoofing'

    # Anomalías generales: llamadas frecuentes en la madrugada o desde roaming
    if row['es_madrugada'] == 1 or (row['hora'] < 6 and row['costo_usd'] > 0.5):
        return 'Anomalía General'

    return 'Otro tipo de fraude'
```

[ ]:
```
data['tipo_fraude_detectado'] = data.apply(clasificar_tipo_fraude_avanzado,␣
 ↪axis=1)
```

[ ]:
```
fraude_counts = data[data['es_fraude'] == 1]['tipo_fraude_detectado'].
 ↪value_counts()

plt.figure(figsize=(5, 2))
fraude_counts.plot(kind='bar', color='tomato', edgecolor='black')
```

```
plt.title('Distribución de Tipos de Fraude Detectados')
plt.xlabel('Tipo de fraude')
plt.ylabel('Número de casos')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

[ ]:
```
df_fraudes = data[data['es_fraude'] == 1][['usuario_id', 'numero_destino',␣
↪'costo_usd', 'tipo_fraude_detectado']]
```

[ ]:
```
# Filtrar fraudes únicamente
df_fraudes = data[data['es_fraude'] == 1].copy()

# Seleccionar variables relevantes (puedes ajustar según tu caso)
features = ['duracion_seg', 'costo_usd', 'hora', 'es_madrugada',␣
↪'llamada_internacional', 'destino_premium']
X = df_fraudes[features]
```

[ ]:
```
scaler = StandardScaler()
df_fraudes_scaled = scaler.fit_transform(df_fraudes)
```

[ ]:
```
# Elegir número de clusters (puedes ajustar con el codo)
kmeans = KMeans(n_clusters=4, random_state=42)
df_fraudes['clusters'] = kmeans.fit_predict(df_fraudes_scaled)
```

[ ]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(df_fraudes_scaled)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df_fraudes['clusters'], cmap='tab10',␣
↪alpha=0.7)
plt.title('Clusters de Fraudes Detectados (PCA)')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.colorbar(label='Cluster')
plt.grid(True)
plt.tight_layout()
plt.show()
```

[ ]:
```
# Ver estadísticas de cada cluster
df_fraudes.groupby('clusters')[features].mean()
```

[ ]:
```
# Diccionario con las etiquetas por número de cluster
cluster_labels = {
```

```
    0: "Wangiri",
    1: "Fraude General Internacional",
    2: "IRSF (premium)",
    3: "SIM Box / Bypass"
}

# Crear una nueva columna con el nombre del tipo de fraude según el cluster
df_fraudes['tipo_fraude_estimado'] = df_fraudes['clusters'].map(cluster_labels)

# Visualizar los primeros registros etiquetados
print(df_fraudes[['clusters', 'tipo_fraude_estimado']].value_counts())
```

[ ]:
```
# Ejmplos de cada tipo de fraude
for tipo in df_fraudes['tipo_fraude_estimado'].unique():
    print(f"\nEjemplos para: {tipo}")
    display(df_fraudes[df_fraudes['tipo_fraude_estimado'] == tipo].head(3))
```

0.2 Autoencoders

Un autoencoder es una red neuronal que aprende a comprimir (codificar) los datos y luego reconstruirlos.

Se compone de dos partes:

Encoder: Reduce la dimensionalidad.

Decoder: Reconstruye la entrada original.

El error de reconstrucción es la diferencia entre la entrada original y la reconstruida por el autoencoder.

[ ]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam

# Dimensión de entrada
input_dim = df_fraudes_scaled.shape[1]

# Estructura del autoencoder
input_layer = Input(shape=(input_dim,))
encoded = Dense(8, activation='relu')(input_layer)
encoded = Dense(4, activation='relu')(encoded)
decoded = Dense(8, activation='relu')(encoded)
decoded = Dense(input_dim, activation='linear')(decoded)

autoencoder = Model(inputs=input_layer, outputs=decoded)
autoencoder.compile(optimizer=Adam(learning_rate=0.001), loss='mse')
```

[ ]:
```
# Entrenar el modelo
history = autoencoder.fit(df_fraudes_scaled, df_fraudes_scaled,
                          epochs=50,
                          batch_size=256,
                          shuffle=True,
                          validation_split=0.1,
                          verbose=1)
```

[ ]:
```
# Calcular el error de reconstrucción
# Reconstrucción
X_pred = autoencoder.predict(df_fraudes_scaled)
reconstruction_error = np.mean(np.power(df_fraudes_scaled - X_pred, 2), axis=1)

# Añade el error al DataFrame
df_fraudes['reconstruction_error'] = reconstruction_error
```

[ ]:
```
# Detección de anomalias (threshold = 95)
threshold = np.percentile(reconstruction_error, 95)
df_fraudes['fraude_autoencoder'] = (reconstruction_error > threshold).
 ↪astype(int)
```

Si el error de reconstrucción es bajo, el patrón es normal → fraude_autoencoder = 0.

Si el error es alto, significa que los datos no se parecen a los patrones que el autoencoder aprendió → posible fraude → fraude_autoencoder = 1.

[ ]:
```
df_fraudes.head()
```

[ ]:
```
print(df_fraudes[['clusters','tipo_fraude_estimado','fraude_autoencoder']].
 ↪value_counts())
```

[ ]:
```
# Conteo de fraudes detectados por cluster
pd.crosstab(df_fraudes['clusters'], df_fraudes['fraude_autoencoder'],
            rownames=['Cluster'],
            colnames=['Fraude Detectado por Autoencoder'])
```

[ ]:
```
# Agrupar para gráfico
grouped = df_fraudes.groupby(['clusters', 'fraude_autoencoder']).size().
 ↪unstack(fill_value=0)

# Gráfico
grouped.plot(kind='bar', stacked=True, figsize=(8,5), colormap='coolwarm')
plt.title('Detección de Anomalías por Cluster')
plt.ylabel('Número de Muestras')
plt.xlabel('Cluster')
plt.legend(title='Autoencoder\nFraude', labels=['No', 'Sí'])
plt.tight_layout()
plt.show()
```

[ ]:
```
inconsistencias = pd.crosstab(
    df_fraudes['tipo_fraude_estimado'],
    df_fraudes['fraude_autoencoder'],
    rownames=['Tipo de Fraude Estimado'],
    colnames=['Detectado por Autoencoder']
)

print(inconsistencias)
```

[ ]:
```
# Filtrar casos donde había un tipo estimado pero el autoencoder no lo detectó
no_detectado = df_fraudes[
    (df_fraudes['tipo_fraude_estimado'].notna()) &
    (df_fraudes['fraude_autoencoder'] == 0)
]

print(f"Número de fraudes estimados NO detectados por el autoencoder:␣
 ↪{len(no_detectado)}")
```

[ ]:
```
# Filtrar casos donde el autoencoder marcó como fraude pero no hay tipo estimado
falsos_positivos = df_fraudes[
    (df_fraudes['tipo_fraude_estimado'].isna()) &
    (df_fraudes['fraude_autoencoder'] == 1)
]

print(f"Número de fraudes detectados sin estimación previa:␣
 ↪{len(falsos_positivos)}")
```

[ ]:
```
# Ver algunos ejemplos de inconsistencias
print(no_detectado[['usuario_id', 'tipo_fraude_estimado', 'duracion_seg',␣
 ↪'costo_usd']].head())
```

0.3 Isolation Forest

[ ]:
```
from sklearn.ensemble import IsolationForest

outliers_df = df_fraudes.copy()
outliers_df['is_outlier'] = 0  # por defecto no es outlier

# Recorremos cada cluster individualmente
for cluster in outliers_df['clusters'].unique():
    subset = outliers_df[outliers_df['clusters'] == cluster]

    # Seleccionamos solo las columnas numéricas relevantes
    features = subset[['duracion_seg', 'costo_usd', 'hora',
                       'es_madrugada', 'llamada_internacional',␣
 ↪'destino_premium']]
```

```
    # Aplicamos Isolation Forest solo dentro de este cluster
    iso_forest = IsolationForest(contamination=0.05, random_state=42)
    preds = iso_forest.fit_predict(features)

    # -1 es outlier, 1 es normal → lo convertimos a 1 y 0
    outliers_df.loc[subset.index, 'is_outlier'] = (preds == -1).astype(int)

# Ahora puedes ver qué filas dentro de cada cluster se consideran outliers
outliers_detectados = outliers_df[outliers_df['is_outlier'] == 1]
```

0.3.1 Analisis inmediato de patrones

[ ]:
```
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')

# Análisis por cluster - ¿qué clusters generan más outliers?

outliers_por_cluster = outliers_detectados.groupby('clusters').agg({
    'is_outlier': ['count'],
    'costo_usd': ['mean', 'max', 'std'],
    'duracion_seg': ['mean', 'max', 'std']
}).round(2)

outliers_por_cluster.columns = ['total_outliers', 'costo_medio', 'costo_max',␣
 ↪'costo_std',
                                'duracion_media', 'duracion_max',␣
 ↪'duracion_std']

# Calcular porcentaje de outliers por cluster
cluster_totals = outliers_df.groupby('clusters').size()
outliers_por_cluster['pct_outliers'] = (outliers_por_cluster['total_outliers'] /
 ↪ cluster_totals * 100).round(2)

print(outliers_por_cluster.sort_values('pct_outliers', ascending=False))
```

[ ]:
```
# Top características que más diferencian
print("\nTOP DIFERENCIAS MÁS SIGNIFICATIVAS:")
top_diferencias = comparacion.reindex(comparacion['diferencia_pct'].abs().
 ↪sort_values(ascending=False).index)
print(top_diferencias.head())
```

[ ]:
```
# ================================
# DETECCIÓN DE NUEVOS PATRONES DE FRAUDE
# ================================

# Outliers con combinaciones sospechosas
```

```
print("\nPATRONES SOSPECHOSOS (Madrugada + Internacional + Alto costo):")
patrones_sospechosos = outliers_detectados[
    (outliers_detectados['es_madrugada'] == 1) &
    (outliers_detectados['llamada_internacional'] == 1) &
    (outliers_detectados['costo_usd'] > outliers_detectados['costo_usd'].
 ↪quantile(0.9))
]
print(f"Casos encontrados: {len(patrones_sospechosos)}")
if len(patrones_sospechosos) > 0:
    print("\nResumen de estos casos:")
    print(patrones_sospechosos[['clusters', 'duracion_seg', 'costo_usd',␣
 ↪'hora']].describe())
```

[ ]:
```
# Análisis de correlaciones anómalas
print("\nCORRELACIONES EN OUTLIERS vs DATOS NORMALES:")
correlaciones_outliers = outliers_detectados[['duracion_seg', 'costo_usd',␣
 ↪'hora']].corr()
correlaciones_normales =␣
 ↪outliers_df[outliers_df['is_outlier']==0][['duracion_seg', 'costo_usd',␣
 ↪'hora']].corr()

print("Correlaciones en OUTLIERS:")
print(correlaciones_outliers.round(3))
print("\nCorrelaciones en datos NORMALES:")
print(correlaciones_normales.round(3))

print("\nDIFERENCIAS en correlaciones:")
diferencias_corr = (correlaciones_outliers - correlaciones_normales).round(3)
print(diferencias_corr)
```

[ ]:
```
# Patrones por cluster
print("\nANÁLISIS DETALLADO POR CLUSTER:")
for cluster in sorted(outliers_detectados['clusters'].unique()):
    cluster_outliers = outliers_detectados[outliers_detectados['clusters'] ==␣
 ↪cluster]
    if len(cluster_outliers) > 0:
        print(f"\n--- CLUSTER {cluster} ---")
        print(f"Outliers: {len(cluster_outliers)}")
        print(f"Costo promedio: ${cluster_outliers['costo_usd'].mean():.2f}")
        print(f"Duración promedio: {cluster_outliers['duracion_seg'].mean():.
 ↪0f} seg")
        print(f"% Madrugada: {cluster_outliers['es_madrugada'].mean()*100:.
 ↪1f}%")
        print(f"% Internacional: {cluster_outliers['llamada_internacional'].
 ↪mean()*100:.1f}%")
```

```
        print(f"% Premium: {cluster_outliers['destino_premium'].mean()*100:.
 ↪1f}%")
```

[ ]:
```
# ================================
# SISTEMA DE SCORING REFINADO
# ================================

# Crear un score compuesto
outliers_df['anomaly_score'] = 0

print("Calculando scores de anomalía por cluster...")
for cluster in outliers_df['clusters'].unique():
    subset = outliers_df[outliers_df['clusters'] == cluster]
    features = subset[['duracion_seg', 'costo_usd', 'hora', 'es_madrugada',␣
 ↪'llamada_internacional', 'destino_premium']]

    iso_forest = IsolationForest(contamination=0.05, random_state=42)
    iso_forest.fit(features)

    # decision_function da scores continuos (más negativo = más anómalo)
    scores = iso_forest.decision_function(features)
    outliers_df.loc[subset.index, 'anomaly_score'] = scores

print("Scores calculados!")
```

[ ]:
```
# Priorizar por score más extremo
print("\nTOP 10 CASOS MÁS ANÓMALOS:")
outliers_priorizados = outliers_df.sort_values('anomaly_score')[['clusters',␣
 ↪'anomaly_score', 'costo_usd', 'duracion_seg', 'is_outlier', 'es_madrugada',␣
 ↪'llamada_internacional']]
print(outliers_priorizados.head(10))
```

[ ]:
```
# Score statistics
print("\nESTADÍSTICAS DE ANOMALY SCORES:")
score_stats = pd.DataFrame({
    'outliers': outliers_df[outliers_df['is_outlier']==1]['anomaly_score'].
 ↪describe(),
    'normales': outliers_df[outliers_df['is_outlier']==0]['anomaly_score'].
 ↪describe()
}).round(4)
print(score_stats)
```

[ ]:
```
# Distribución de scores por cluster
print("\nDISTRIBUCIÓN DE SCORES POR CLUSTER:")
score_by_cluster = outliers_df.groupby('clusters')['anomaly_score'].
 ↪agg(['mean', 'std', 'min', 'max', 'count']).round(4)
print(score_by_cluster)
```

[ ]:
```
# ================================
# VALIDACIÓN Y FEEDBACK
# ================================

# Casos más críticos para revisión manual
print("\nTOP 10 CASOS CRÍTICOS PARA REVISIÓN MANUAL:")
casos_criticos = outliers_df.nsmallest(10, 'anomaly_score')[
    ['clusters', 'anomaly_score', 'costo_usd', 'duracion_seg', 'hora',␣
 ↪'es_madrugada', 'llamada_internacional', 'destino_premium']
]
print(casos_criticos)
```

[ ]:
```
# Métricas por cluster para ajustar contamination
print("\n  MÉTRICAS POR CLUSTER (para ajustar parámetros):")
metricas_cluster = outliers_df.groupby('clusters').agg({
    'is_outlier': ['mean', 'sum'],  # % y total de outliers por cluster
    'anomaly_score': ['mean', 'std', 'min'],
    'costo_usd': ['mean', 'max'],
    'duracion_seg': ['mean', 'max']
}).round(4)

metricas_cluster.columns = ['pct_outliers', 'total_outliers', 'score_medio',␣
 ↪'score_std', 'score_min',
                            'costo_medio', 'costo_max', 'duracion_media',␣
 ↪'duracion_max']
print(metricas_cluster)
```

[ ]:
```
# Recomendaciones de ajuste
print("\nRECOMENDACIONES DE AJUSTE:")
for cluster in metricas_cluster.index:
    pct = metricas_cluster.loc[cluster, 'pct_outliers']
    if pct > 0.1:  # Más del 10% son outliers
        print(f"Cluster {cluster}: {pct:.1%} outliers - Considerar reducir␣
 ↪contamination")
    elif pct < 0.02:  # Menos del 2% son outliers
        print(f"Cluster {cluster}: {pct:.1%} outliers - Considerar aumentar␣
 ↪contamination")
    else:
        print(f"Cluster {cluster}: {pct:.1%} outliers - Configuración adecuada")
```

[ ]:
```
# Matriz de confusión simulada (si tuvieras labels reales)
print("\nSIMULACIÓN DE EFFECTIVENESS:")
# Asumiendo que los casos con scores muy bajos son más probables de ser fraude␣
 ↪real
threshold_critico = outliers_df['anomaly_score'].quantile(0.05)
casos_muy_anomalos = (outliers_df['anomaly_score'] < threshold_critico).sum()
print(f"Casos con score extremadamente bajo (<5%): {casos_muy_anomalos}")
```

```
print(f"Estos representan el {casos_muy_anomalos/len(outliers_df)*100:.2f}% del␣
  ↪total")
```